ISSN 0860-1941

CENA 300 zł

mikroklan

vademecum techniki mikrokomputerowej 1'87

Komputery
IBM PC/XT/AT
i
kompatybilne

WordStar 2000

Konwerter
Centronics RS-232C

Interfejs do Atari

WYDAWNICTWO NOT SIGMA

Rozpoczynamy czwarty już rok, miejmy nadzieję, regularnego tym razem wydawania MIKROKLANU. Zachęceni wielkim sukcesem pierwszych dwóch lat, gdy był on zaledwie skromną ośmiostronicową wkładką do miesięcznika INFORMATYKA, podjęliśmy w ubiegłym roku próbę wydawania tego tytułu jako samodzielnego czasopisma. Liczne trudności w realizacji tego przedsięwzięcia spowodowały, że 1986 rok zdołaliśmy zamknąć wydaniem, niestety, zaledwie trzech zeszytów.

Warto przypomnieć, że rozpoczynaliśmy niewątpliwie jako pierwsi i jedyni na krajowym rynku wydawniczym propagatorzy techniki mikrokomputerowej. W ubiegłym roku sytuacja uległa jednak radykalnej zmianie: pojawili się inni wydawcy wielkonakładowych czasopism mikrokomputerowych. W konsekwencji pierwotne założenia programowe MIKROKLANU należało zmienić w sposób zasadniczy. Szybko rosnący poziom wiadomości i praktycznych doświadczeń posiadaczy i użytkowników mikrokomputerów w Polsce, a zwłaszcza rozwój znajomości sprzętu profesjonalnego i jego zastosowań, wymaga stopniowego przekształcania MIKROKLANU w czasopismo ukierunkowane na taką właśnie problematykę. Powinien on zaspokajać potrzeby informacyjne i samokształceniowe nie tylko kilkunastotysięcznej grupy zaawansowanych użytkowników sprzętu profesjonalnego, ale również przygotowywać i stymulować ogromny rynek przyszłych, a tak potrzebnych krajowi masowych zastosowań komputerów we wszystkich rodzajach stanowisk pracy.

Oprócz prezentowania krajowych pomysłów i osiągnięć w dziedzinie wszystkich rodzajów konstrukcji, oprogramowania i zastosowań, MIKROKLAN w znacznym stopniu będzie zawierać bezpośrednie przedruki najciekawszych publikacji z wielkonakładowych czasopism zagranicznych: zachodnioniemieckiego MICRO oraz austriackiego OUTPUT. W sytuacji bardzo ograniczonego dopływu do kraju literatury zagranicznej, przedruki te zapewnią bezpośrednie, i co najistotniejsze, aktualne i rzeczowe informowanie opinii publicznej o światowym postępie w technice mikrokomputerowej i jej zastosowaniach. Przyczyni się to również do tak potrzebnego kształtowania świadomości społecznej oraz konkretnej wiedzy na temat konieczności, efektywności i rzeczywistych możliwości zastosowań mikrokomputerów.

Będziemy starali się dbać o konkretność publikacji. Wychodzimy z założenia, że wobec stosunkowo wysokiej, a całkowicie niezależnej od nas ceny MIKROKLANU, rację bytu będą miały przede wszystkim materiały o dużych walorach praktycznych. Będziemy dbali o to, aby każdy ze stutysięcznej rzeszy nabywców naszego czasopisma mógł znaleźć informacje, których pilnie poszukuje, a których nie znajdzie w tak obszernym już krajowym czasopiśmiennictwie mikrokomputerowym.

Jest to zadanie bardzo trudne i dlatego w kształtowaniu profilu MIKROKLANU powinien aktywnie uczestniczyć praktycznie każdy Czytelnik. Otrzymaliśmy już wiele listów zawierających cenne uwagi i propozycje, które na bieżąco analizujemy i staramy się jak najszybciej uwzględnić w kolejnych numerach. Wierzymy, że zaangażowanie osobiste, jakie obserwujemy w środowisku użytkowników sprzętu mikrokomputerowego, będzie przenosiło się również na naszą działalność redakcyjną i wpływało na coraz lepsze, zgodne z potrzebami najszerszych kręgów środowiska, kształtowanie treści MIKROKLANU. Jedynie tą drogą staniemy się rzeczywiście Waszym czasopismem, którego kolejnych numerów będziecie oczekiwać z autentyczną niecierpliwością. Powtarzamy więc nasz apel z poprzedniego numeru: kształtujmy wspólnie treść MIKROKLANU!

Redakcja

mikroklan 1 '87

Wprowadzenie
w zagadnienia
techniki
mikrokomputerowej

Redaguje zespół miesięcznika INFORMATYKA: Maciej Adamczyk, Piotr Breitkopf, Teresa Jabłońska (sekretarz redakcji), Władysław Klepacz (red. naczelny), Krzysztof Kontek, Paweł Mijal, Jerzy Orkiszewski, Piotr Parlewicz, Maria Pawlak (sekretarz redakcji), Zbigniew Pojmański, Danuta Sot, Krzysztof Rzymkowski, Romuald Szuniewicz
Józef Kaczmarczyk (opracowanie graficzne)

WYDAWNICTWO CZASOPISM I KSIĄŻEK TECHNICZNYCH

SIGMA

PRZEDSIĘBIORSTWO NACZELNEJ ORGANIZACJI TECHNICZNEJ

00-950 Warszawa skrytka 1004
ul. Biała 4

ISSN 0860-1941
Wydawnictwo Czasopism
i Książek Technicznych
NOT-SIGMA
Warszawa 1987

Drukowano na zlecenie ARS POLONA

Redakcja: 01-517 Warszawa,
ul. Mickiewicza 18 m. 17, tel. 39 14 34
Skład: technika fotoskładu Eurocat 150
Wydawnictwo NOT-Sigma
Druk: Bohmann Druck und Verlag
GmbH&Co. KG, Wiedeń. Austria
Nakład 100 000 egz. K-80
Cena 300 zł

SPIS TREŚCI

SPRZĘT

- Konwerter Centronics RS-232C
  Stanisław Błeszyński 20
- Układ interfejsu do Atari 600XL i 800XL
  Mieczysław Szczęch, Krzysztof Skonieczny 32

OPROGRAMOWANIE

- Mniej znane możliwości asemblera GENS3
  Piotr Parlewicz 15
- WordStar 2000, czyli nowe szaty króla
  micro 22
- Czym jest UNIX (4). Powłoka
  Janusz Zalewski 27
- Elementy grafiki w języku TURBO-Pascal
  Jan Bielecki 30

ZASTOSOWANIA

- AutoCAD
  micro 9

TEST

- IBM PC/AT
  micro 7
- Jakość dyskietek (1)
  micro 12

INFORMATYCZNE ABC

- Uczmy się Pascala (1)
  micro 2
- BASIC dla początkujących (4)
  micro 24

OPIS

- Komputery IBM PC/XT/AT
  Krzysztof Kontek 4
- Spokrewnione z wyboru, czyli kompatybilne z własnej woli
  micro 17

UCZMY SIĘ PASCALA!

W ostatnich latach można obserwować stały wzrost znaczenia języka Pascal. Ponieważ był on początkowo zaprojektowany do potrzeb dydaktycznych, jest szczególnie atrakcyjny dla osób pragnących opanować umiejętność poprawnego i efektywnego programowania. Używanie Pascala wyrabia dobre nawyki ścisłego myślenia i umożliwia pisanie prawidłowo skonstruowanych i łatwo zrozumiałych programów. Rozpoczynamy obecnie druk kursu Pascala w odcinkach. Jego treść jest podana w sposób nie wymagający uprzedniej znajomości podstaw programowania.

Język Pascal opracował w 1971 r. Niklaus Wirth, profesor informatyki Politechniki w Zurychu. Chociaż zaprojektowany w założeniu do potrzeb dydaktycznych, Pascal stał się obok BASICA najbardziej popularnym językiem programowania mikrokomputerów. Jego początkowe założenia mają szczególne znaczenie dla początkujących użytkowników, ponieważ Pascal narzuca logiczną i dobrze zorganizowaną strukturę programu.

Najsłabszą stroną początkowej wersji Pascala jest uboga obsługa plików rezydujących w pamięci zewnętrznej. W związku z tym w poszczególnych implementacjach przyjęto różne sposoby rozwiązania tego problemu, rozszerzając pierwotny zbiór instrukcji.

Negatywnym odbiciem tego faktu jest niedokładne przestrzeganie normy Pascala, z czego wynika, że program napisany w jednej wersji musi być dla innej odpowiednio zmieniony.

Prezentowany kurs opiera się na czterech różnych implementacjach (Pascal MT+, UCSD/Apple Pascal, MS-Pascal oraz TURBO-Pascal), które łącznie pokrywają zapotrzebowanie znacznej części eksploatowanych obecnie mikrokomputerów. Tam, gdzie poszczególne wersje wykazują różnice, zostanie to odpowiednio zaznaczone.

W odróżnieniu od interpretowanego języka BASIC program w Pascalu przed jego wykonaniem musi zostać przekształcony za pomocą specjalnego programu zwanego kompilatorem albo do postaci bezpośrednio zrozumiałej przez procesor (Pascal MT+, TURBO-Pascal), albo też kodu pośredniego (ang. p-code), interpretowanego podczas wykonywania programu (UCSD-Pascal, MS-Pascal). Dla użytkownika różnice te nie mają jednak większego znaczenia.

Praktyczna nauka języka wymaga dysponowania mikrokomputerem wyposażonym w co najmniej jeden napęd dyskietek oraz kompilator Pascala.

Przed rozpoczęciem programowania, należy zaznajomić się ze sposobem wprowadzania tekstów do komputera, ich zapamiętywania na dyskietce, ponownego wprowadzania do pamięci oraz nanoszenia zmian. Potrzebny do tego edytor istnieje w wielu wersjach Pascala; można również posłużyć się w tym celu programem przetwarzania tekstów, takim jak np. WordStar. A teraz pierwszy przykład programu:

```
PROGRAM pierwszy_program;
  (*jest to nasz pierwszy przykład programu*);
BEGIN
  WRITELN ('uczę się programować');
END.
```

Program w Pascalu rozpoczyna się słowem zastrzeżonym *PROGRAM*. Po nim następuje, przedzielona spacją, nazwa programu. Wiersz ten, nazywany nagłówkiem programu, zamykamy średnikiem.

Drugi wiersz naszego przykładowego programu jest komentarzem, który kompilator ignoruje. Komentarze umieszczamy w dowolnym miejscu programu zamykając je w nawiasach klamrowych lub w zestawie: nawias zwykły i gwiazdka. Przy tej metodzie zapisu można włączyć do tekstu programu uwagi pomagające zrozumieć intencję programisty lub cel wprowadzonych zmian (patrz „Uwagi do składni").

Właściwy program, a więc jego część instrukcyjną, rozpoczyna słowo *BEGIN*. Wiersz *WRITELN* ('uczę się programować') jest w tym przykładzie jedyną instrukcją. *WRITELN* jest skrótem angielskiego rozkazu 'Write a line' – po polsku: *pisz wiersz*. Komputer otrzymuje rozkaz wyprowadzenia jednego wiersza na ekran. W tym celu musi on wiedzieć, co ma zawierać ten wiersz. Tego rodzaju parametry są umieszczone w Pascalu zawsze w nawiasach okrągłych. Dzięki apostrofom, które ograniczają zdanie *'uczę się programować'*, kompilator rozpoznaje, że występuje tu ciąg znaków. Zapisuje go na ekranie i ustawia kursor na początku następnego wiersza.

Po każdej zakończonej instrukcji znajduje się średnik. Sytuacje, w których można opuścić średnik zostaną omówione później.

Zamiast *WRITELN* można użyć również instrukcji *WRITE*. Wykonanie instrukcji będzie identyczne, jednak kursor ustawi się teraz po ostatniej wyprowadzonej literze, tak aby można było dopisać dalsze słowa w tym samym wierszu.

Słowo *END*, zakończone kropką, oznacza koniec programu.

Po wprowadzeniu do komputera za pomocą edytora tekstu naszego przykładowego programu wywołujemy kompilator, który przekształci tekst źródłowy na właściwy program. Procedura ta jest niezbędna po każdej zmianie lub wprowadzeniu nowego programu.

UWAGI DO SKŁADNI I SPOSOBU ZAPISU

Wszystkie słowa zaznaczone w programach tłustym drukiem, np. **BEGIN** mają w Pascalu określone, niezmienne znaczenie. Nazywane są one słowami zastrzeżonymi i mogą być w zależności od potrzeby pisane dużymi lub małymi literami.

Słowa pisane dużymi literami ale drukiem normalnym, mają w Pascalu określone znaczenie. Znaczenie to może być jednak zmienione metodami, które zostaną później wyjaśnione. Przykładem jest tu *WRITELN*.

Słowa pisane małymi literami są dobierane w sposób dowolny i definiowane przez programistę. Jest celowe, aby je nazywać zgodnie z ich znaczeniem, gdyż dzięki temu programy stają się bardziej przejrzyste. Powinny one zawierać tylko litery i cyfry, przy czym pierwszym znakiem musi być litera. Jedyny pojedynczy dopuszczalny znak przestankowy – znak podkreślenia – służy do optycznego dzielenia nazw wieloczłonowych.

Należy zaznaczyć, że jedną z cech różniących kompilatory jest liczba rozpoznawanych znaków nazw zmiennych. Jeżeli np. kompilator rozpoznaje tylko 13 pierwszych znaków, to wprawdzie wyrażenie *słowoprogramu1* z jego czternastoma znakami będzie dopuszczalne, ale kompilator nie odróżni go od wyrażenia *słowoprogramu2*.

Teksty, które mają być wprowadzone na ekran lub drukarkę zaczynają się i kończą apostrofem.

Komentarze są wyłącznie uwagami programisty ułatwiającymi późniejsze zrozumienie treści programu. Nie mają one żadnego wpływu na jego wykonanie oraz są ignorowane przez kompilator. Są one ograniczone przez nawiasy klamrowe lub nawiasy okrągłe z gwiazdkami.

Znak nowego wiersza w tekście programu jest traktowany jako separator, podobnie jak spacja lub tabulator. Dopuszczalne jest rozbicie pojedynczej instrukcji na kilka wierszy lub pisanie w jednym wieazu więcej niż jednej instrukcji. Pozwala to nadać tekstom programów bardziej przejrzystą szatę graficzną.

Jeżeli kompilator wyprowadzi komunikat o błędzie, powstałym przy wprowadzaniu tekstu z klawiatury, to w zależności od wersji Pascala nastąpi albo automatyczne wywołanie edytora z tekstem programu, a kursor przeskoczy na błędną pozycję, albo też procedurę tę trzeba będzie wykonać ręcznie. Od momentu bezbłędnej kompilacji programu, można go wykonywać i sprawdzać.

Istotną nowością w programie *dialog* jest wprowadzenie zmiennych. Pod pojęciem tym należy rozumieć dane, których dokładna wartość będzie ustalona dopiero wtedy, gdy program będzie realizowany. Są one istotnym składnikiem każdego języka programowania.

W odróżnieniu od BASICA, w Pascalu wszystkie używane zmienne muszą być zadeklarowane przed ich użyciem w nagłówku, tj. w początkowej części programu poprzedzającej instrukcję. Zmienną deklarujemy po słowie *VAR* przez podanie jej nazwy oraz typu, tzn. zakresu wartości, które zmienna może przyjmować. Od typu zmiennej zależy rodzaj operacji, jakie można na niej wykonywać. Nazwa zmiennej może być wybrana dowolnie, ale musi zaczynać się od litery (patrz „Uwagi do składni").

W przykładzie nazwa zmiennej brzmi *znak*. Oddzielony od nazwy dwukropkiem następuje typ danych – w przykładzie *CHAR*. Jest to skrót angielskiego słowa „character" (znak pisarski). Zmienna typu *CHAR* może reprezentować każdy znak kodu ASCII, w szczególności cyfrę, literę lub znak przestankowy.

Dalsze typy danych stosowane w Pascalu są zawarte w załączonej tabeli. W następnych odcinkach kursu zostanie wyjaśnione, jak samemu można definiować typy danych.

TYPY DANYCH W PASCALU

Nazwa	Opis	Przykład
CHAR	Znaki kodu ASCII: litera, znak przestankowy, cyfra i inne	a A 1 8 $; i
INTEGER	Liczba całkowita (bez miejsca po przecinku), z reguły pomiędzy -32 768 i +32 767	457-28954
BYTE	liczba całkowita pomiędzy 0 i 255. Typ danych występujących tylko w TURBO-Pascalu	1 58 217 0
REAL	Liczba z miejscami po przecinku. Należy zauważyć, że miejsca po przecinku są oddzielane kropką (amerykański sposób zapisu)	2.24756 215.1
STRING	Tekst do 255 znaków	
BOOLEAN	Zmienna logiczna. Zmienna tego typu może przybierać tylko jedną z dwóch wartości: prawdy lub fałszu (TRUE, FALSE). Służy ona do sterowania przebiegiem programu.	

Dalsze szczegóły oraz przykłady zastosowania różnych typów danych pojawią się w następnych odcinkach kursu.

Nasz program powinien wyprowadzić na ekran wiersz *uczę się programować*. Aby zaznajomić się ze sposobem działania, można ten przykład zmodyfikować tak, aby wyprowadzić inny tekst. Przez proste wstawienie dodatkowego wiersza, np:

```
WRITE ('uczę się programować');
WRITELN ('sprawia to mi przyjemność!');
```

można wyprowadzić na ekran również dłuższe teksty oraz zbadać różnicę pomiędzy instrukcjami *WRITELN* i *WRITE*.

Następny program przedstawia kilka dodatkowych możliwości języka:

```
PROGRAM dialog;
VAR znak: CHAR;
BEGIN
  WRITE ('wprowadź jeden znak');
  READLN (znak);
  WRITELN ('wprowadziłeś znak,' znak);
END
```

Znaczenie wiersza:

```
WRITE (,'wprowadź jeden znak');
```

jest w zasadzie już wyjaśnione – wezwanie *wprowadź jeden znak* zostanie wyprowadzone na ekran, a kursor zatrzyma się po spacji. Następny wiersz:

```
READLN (znak);
```

powoduje, że uprzednio zadeklarowanej zmiennej *znak* przyporządkowana zostanie konkretna, podana przez użytkownika wartość. *READLN* jest skrótem angielskiego określenia „read a line" – po polsku *czytaj wiersz*. Napotykając instrukcję *READ* lub *READLN* komputer oczekuje, że zostanie wprowadzony wiersz danych, w naszym przypadku jeden znak. Wprowadzenie wiersza polega na napisaniu go i naciśnięciu klawisza *RETURN* lub *ENTER*. W wypadku instrukcji *READLN*, kursor zostanie ustawiony na początku nowego wiersza i program będzie kontynuowany od następnej instrukcji. Wykonanie instrukcji *READ* różni się tym, że po naciśnięciu klawisza *RETURN* lub *ENTER* kursor nie zmienia pozycji i pozostaje bezpośrednio po ostatnim wprowadzonym w wierszu znaku. Jest to tylko pozornie skomplikowane, co najlepiej sprawdzić przez przećwiczenie wprowadzania i wyprowadzania różnych wartości zmiennej *'znak'* dla kilku wersji programu, z wykorzystaniem instrukcji *READ*, *READLN*, *WRITE*, *WRITELN*.

Ostatni wiersz przykładowego programu zawiera właściwie dwie instrukcje dla komputera:

```
WRITE ('wprowadziłeś znak');
WRITELN (znak);
```

Ponieważ obie instrukcje powodują wyprowadzenie tekstów na ekran, to można je połączyć. Kolejne argumenty instrukcji *WRITE* są oddzielone przecinkami. Należy zaznaczyć, że zmienna *znak* nie jest ograniczona apostrofami, ponieważ nie interpretuje się jej jako tekstu przeznaczonego do wypisania na ekranie, lecz żąda podania wartości zmiennej, w naszym wypadku jednego znaku.

Reasumując, program *dialog* realizuje następujące działania:

- wezwanie użytkownika do wprowadzenia jednego znaku,
- przyporządkowanie tego znaku zmiennej *znak*,
- wyprowadzenie wiersza z wartością tej zmiennej łącznie z ustalonym tekstem.

Przykładowi brakuje jednak istotnego elementu programu komputerowego: wprowadzony znak nie zostanie zmieniony, lecz ponownie wyprowadzony w identycznej postaci. W następnej części pokażemy, jak można przetwarzać zmienne za pomocą programu.

POWTORZENIE NAJWAŻNIEJSZYCH POJĘĆ

BEGIN	początek części instrukcyjnej programu.
CHAR	patrz „Typy danych w Pascalu"
END.	koniec części instrukcyjnej programu (uwaga na kropkę)
Kompilator	program tłumaczący tekst napisany w Pascalu na kod wewnętrzny komputera.
Nagłowek programu	każdy program rozpoczyna się słowem **PROGRAM**, po którym następuje jego nazwa; w dalszej części nagłówka znajduje się deklaracja zmiennych.
READ, READLN	instrukcje wprowadzania danych.
VAR	początek deklaracji zmiennych.
WRITE, WRITELN	rozkazy wyprowadzania tekstu na ekran. *WRITELN* ustawia ponadto kursor na początku następnego wiersza.
Zmienna	dowolnie przyjęta nazwa, reprezentująca pojęcie, którego zasadnicze znaczenie oraz typ – jednak bez dokładnej wartości – zostały z góry deklarowane.

KOMPUTERY IBM PC/XT/AT

Firma IBM, specjalizująca się do końca lat siedemdziesiątych głównie w produkcji dużych komputerów, bardzo długo zwlekała z wprowadzeniem na rynek swojego komputera osobistego. Dopiero w połowie 1980 roku, gdy takie firmy jak Apple czy Radio Shack miały już milionowe obroty, zdecydowano się stworzyć dwunastoosobową grupę, której zadaniem stało się opracowanie konkurencyjnego modelu. Dokładnie po roku (!) bardzo intensywnej pracy (praktycznie bez dni wolnych) komputer IBM PC (Personal Computer) był gotów. Mało kto spodziewał się wówczas, że odniesie on tak wielki sukces handlowy.

Po wprowadzeniu na rynek w marcu 1983 roku ulepszonej wersji komputera oznaczonej literami XT (eXTended), firma IBM wysunęła się na pierwsze miejsce wśród producentow minikomputerów i pozycję tę utrzymała po ukazaniu się w połowie 1984 roku jeszcze nowocześniejszej wersji IBM AT (Advanced Technology). Do chwili obecnej na całym świecie sprzedano już kilka milionów komputerów IBM PC/XT/AT. Stał się on standardem wśród komputerów osobistych i profesjonalnych, o czym świadczy fakt, że wiele firm opracowuje swoje nowe modele tak, aby były z nim zgodne, czyli kompatybilne.

Począwszy od połowy 1984 roku niewielka liczba egzemplarzy tego komputera zaczęła docierać także do Polski, głównie dzięki firmom polonijnym i przedsiębiorstwom zagranicznym. Opracowaniem komputera zgodnego z modelem IBM PC/XT zajęły się także przedsiębiorstwa państwowe (spółka „Mikrokomputery"). Coraz więcej sprzętu trafia do rąk prywatnych, ze względu na szybko malejące ceny, zwłaszcza w krajach Dalekiego Wschodu. Komputer z pamięcią 640 KB, dwoma napędami dysków i płytką wielofunkcyjną można kupić już za 550 dolarów, a w najprostszej konfiguracji bez dysków za 300 dolarów. Należy się spodziewać, że w najbliższym czasie komputer IBM PC/XT/AT zdominuje w Polsce rynek komputerów profesjonalnych.

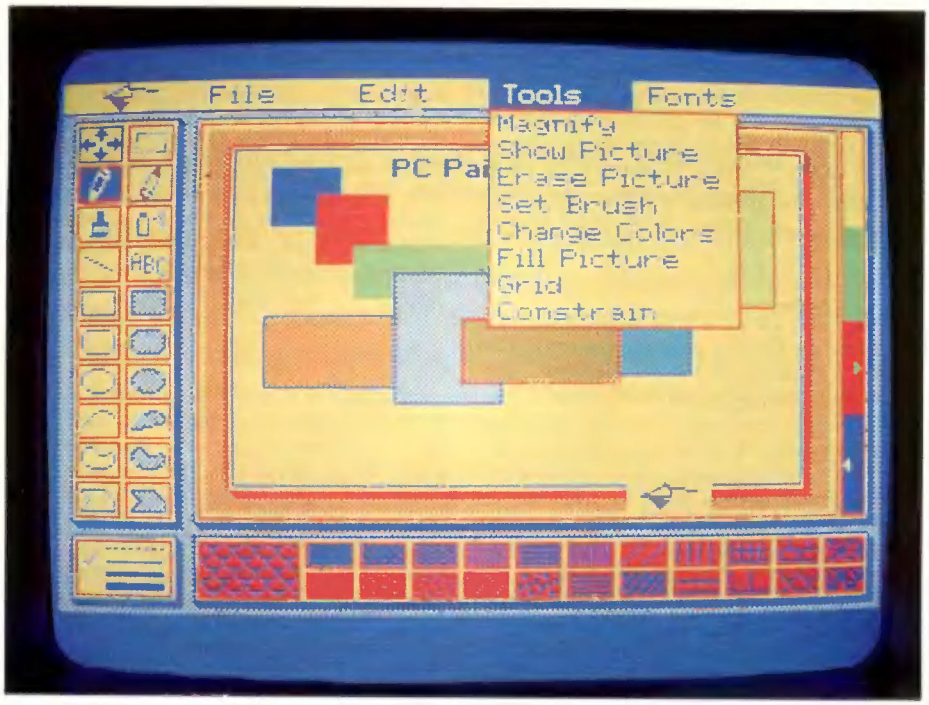
W komputerze IBM PC/XT zastosowano mikroprocesor Intel 8088, który pojawił się na rynku już 10 lat temu, nieco później po mikroprocesorze 8086. Obydwa mikroprocesory mają taką samą 16-bitową architekturę wewnętrzną. Różnią się tylko tym, że szyna danych mikroprocesora 8086 jest 16-bitowa, a mikroprocesora 8088 8-bitowa. Z tego względu mikroprocesor 8088 wykonuje przesłania danych w dwóch etapach, co powoduje wolniejsze wykonywanie programów niż w wypadku mikroprocesora 8086. Wyrażany jest czasami kontrowersyjny pogląd, że mikroprocesor 8088 jest w zasadzie szybkim, 8-bitowym mikroprocesorem o rozbudowanej liście instrukcji, wykonywanych na liczbach 16-bitowych.

Dlaczego konstruktorzy komputera IBM PC zdecydowali się na ten właśnie, nienajnowocześniejszy już w 1980 roku mikroprocesor, a nie na dostępne już na rynku mikroprocesory w pełni 16-bitowe, takie jak 8086, czy jeszcze lepsze Motorola 68000 lub Zilog 8000? O wyborze mikroprocesora firmy Intel zadecydowało głównie bardzo bogate oprogramowanie, dostępne dla mikroprocesorów 8080 i 8085, które mogło być prawie automatycznie przeniesione na mikroprocesory 8086 i 8088. Również programy napisane w języku asemblera mikroprocesora Z80 mogły być stosunkowo łatwo przetłumaczone na język asemblera mikroprocesorów firmy Intel.

Za wyborem mikroprocesora z 16-bitową wewnętrzną szyną danych przemawiała chęć zbudowania komputera nowoczesnego, który mógłby zaspokoić potrzeby użytkowników przez całe dziesięciolecie. Z drugiej strony, użycie mikroprocesora z 8-bitową zewnętrzną szyną danych pozwalało na znaczne uproszczenie konstrukcji i obniżenie ceny komputera. Brano tu pod uwagę następujące fakty:

- większość urządzeń zewnętrznych, takich jak: drukarki, przetworniki analogowo-cyfrowe (A/C) i cyfrowo-analogowe (C/A) miała 8-bitową szynę danych,
- układy opracowane dla mikroprocesorów 8080 i 8085, takie jak: generator zegarowy 8284, sterownik magistrali 8288, sterownik przerwań 8259A, sterownik układów we-wy, układ czasowy 8253-5 i sterownik DMA 8237A-5 były już masowo produkowane i w związku z tym ich cena była niska,
- ośmiobitowa organizacja pamięci pozwalała na zmniejszenie kosztu pamięci.

Już w chwili powstania komputer IBM PC nie



był najnowocześniejszą konstrukcją. Jego twórcy starali się stworzyć układ, który za stosunkowo niską cenę dawałby maksymalnie duże możliwości.

Mikroprocesor 8088 użyto również w modelu XT, wbrew przewidywaniom wielu ekspertów oczekujących zastosowania mikroprocesora 80186. Zmiany dokonano dopiero w modelu AT, wykorzystując mikroprocesor Intel 80286. Jest to nowoczesny mikroprocesor 16-bitowy o przestrzeni adresowej 16 MB, z rozbudowaną architekturą wewnętrzną (m.in. z możliwością pracy w trybie wielozadaniowym oraz z zabezpieczeniem zbiorów przydzielonych do konkretnych zadań). Należy jednak podkreślić, że nie jest to mikroprocesor 32-bitowy, co usiłują zasugerować niektórzy sprzedawcy komputerów opartych na tym mikroprocesorze.

W komputerach IBM PC/XT/AT przewidziano znaczne rozszerzenie możliwości funkcjonalnych mikroprocesorów 8088 i 80286, za pomocą koprocesorów arytmetycznych 8087 i 80287. Przyspieszają one wykonywanie operacji matematycznych nawet do 100 razy, co pozwala na użycie tych komputerów nawet do bardzo złożonych obliczeń naukowych.

PAMIĘĆ

Mikroprocesor 8088 ma 20-bitową szynę adresową, co oznacza, że może adresować pamięć o pojemności 1 MB. W komputerze IBM PC/XT można wykorzystywać w zasadzie tylko początkowe 640 KB pamięci. Reszta jest przewidziana na pamięć obrazu, pamięć ROM oraz do przyszłego wykorzystania przez firmę IBM.

Początkowo opisane możliwości były wykorzystane tylko w niewielkim stopniu. Pierwsze egzemplarze komputera IBM PC miały 40 KB pamięci ROM, w której był umieszczony opracowany przez firmę Microsoft interpreter języka BASIC oraz zaledwie 16 KB pamięci RAM, z możliwością jej rozszerzenia do 64 KB na płycie głównej (ang. motherboard). Szybko jednak zastąpiono układy pamięci 4116 o organizacji 16K × 1b układami o organizacji 64K × 1b, co spowodowało znaczne rozszerzenie pamięci RAM.

W najczęściej spotykanej konfiguracji model XT ma 256 KB pamięci RAM na płycie głównej, przy czym istnieje możliwość jej rozszerzenia za pomocą dodatkowej płytki o dalsze 256 KB lub 384 KB, co łącznie daje 640 KB.

Coraz częściej spotyka się płyty główne skonstruowane w ten sposób, że mogą pomieścić całą pamięć 640 KB. Wykorzystuje się w nich układy pamięci 256 Kb (18 sztuk) i 64 Kb (18 sztuk). Układy pamięci mają maksymalny czas dostępu 250 ns. Dla zwiększenia niezawodności systemu cała pamięć RAM ma organizację nie ośmio- lecz dziewięciobitową, przy czym ostatni bit jest bitem parzystości. Oprogramowanie podstawowe komputera IBM PC tzw. BIOS (Basic Input-Output System) znajduje się w pamięci ROM o pojemności 8 KB. Na płycie głównej znajdują się podstawki, w których użytkownik może umieścić dodatkową pamięć EPROM z interpreterem języka BASIC.

Mikroprocesor 80286 wyposażony w 24-bitową szynę adresową może zaadresować pamięć o pojemności 16 MB. W modelu AT przewidziano użycie początkowych 3 MB pamięci. W konfiguracji podstawowej instaluje się tylko 512 lub 640 KB pamięci RAM, pozostawiając użytkownikowi możliwość jej rozszerzenia za pomocą dodatkowych płytek.

PAMIĘĆ MASOWA

Do przechowywania programów i danych w komputerze IBM PC/XT stosuje się dyskietki 5 1/4 cala. W początkowej wersji IBM PC stosowano dwa mechanizmy napędowe dyskietek (ang. drive), które umożliwiały zapis jednostronny z podwójną gęstością w standardzie MFM. Przy 40 ścieżkach, z których każda miała 8 sektorów po 512 bajtów, dawało to 163 840 bajtów na dyskietkę. Szybkość przesyłania danych wynosiła ok. 250 kB na sekundę.

W modelu XT stosuje się napędy pozwalające na zapis dwustronny. Poza tym nowa wersja systemu operacyjnego DOS 2.0 umożliwia sformatowanie na jednej ścieżce nie 8, lecz 9 sektorów. Pozwoliło to na zwiększenie pojemności sformatowanej dyskietki do 360 kB. Warto dodać, że wersja DOS 2.0 automatycznie rozpoznaje, jak dyskietka jest zapisana – jednostronnie czy dwustronnie i czy liczba sektorów wynosi 8 czy 9. Sterownik jest zbudowany na podstawie układu NEC 765, co umożliwia sterowanie także napędów dyskietek 8-calowych. Są one sterowane w sposób czysto programowy, co pozwala na stosunkowo łatwy odczyt dyskietek zapisanych przez inne komputery.

W modelu XT przewidziano też możliwość dołączenia dysku sztywnego typu Winchester o pojemności 10 MB, który pozwala na przechowanie ok. 5 000 stron maszynopisu. Dysk sztywny zawiera dwie ciągle wirujące płyty. Na każdej z czterech powierzchni można umieścić 306 ścieżek o siedemnastu 512-bajtowych sektorach, co daje łącznie 10 653 696 bajtów. Szybkość przesyłania informacji jest o wiele większa niż w wypadku dysków elastycznych i wynosi 5 mln bitów na sekundę. Nie jest ona ograniczona czasem potrzebnym na uzyskanie przez dysk odpowiedniej prędkości obrotowej (tak jak to jest w wypadku dysków elastycznych),

gdyż jego płyty wirują bez przerwy. Dysk sztywny charakteryzuje się poza tym bardzo dużą niezawodnością zapisu i odczytu, gdyż jest szczelnie zamknięty w obudowie, co uniemożliwia przypadkowe uszkodzenie lub zabrudzenie jego powierzchni przez użytkownika.

W modelu AT, a w ostatnim okresie coraz częściej także w modelu XT, stosuje się dyski sztywne o pojemności 20 MB. Poza tym wykorzystuje się jeden mechanizm napędu dysków elastycznych o pojemności 1,2 MB. System operacyjny DOS 3.0 pozwala na odczyt dyskietek zapisanych na modelu XT.

W pierwotnej wersji komputera IBM PC istniała możliwość użycia magnetofonu kasetowego jako pamięci masowej. Spodziewano się, że w ten sposób znajdzie się nabywców, którzy chcąc obniżyć cenę komputera zrezygnują z dyskietek. Przewidywania te nie spełniły się i w modelu XT całkowicie zrezygnowano z wyjścia dla magnetofonu kasetowego.

GRAFIKA I WYJŚCIE WIZYJNE

W komputerze IBM PC/XT obraz wizyjny może być uzyskiwany za pomocą kilku, różnych płytek.

Płytka grafiki kolorowej (ang. color graphic card) opracowana przez firmę IBM pozwala na wypisanie na ekranie tekstu, a także tworzenie obrazów graficznych. Obraz może być wyświetlany w dwóch trybach: 25 linii po 40 znaków lub 25 linii po 80 znaków (w drugim wypadku niezbędny jest monitor RGB). Znaki są zakodowane w matrycy 5×7 punktów, przy czym na jeden znak przypada pole 8×8 punktów. W podstawowym trybie pracy każdy znak może mieć jeden z 16 kolorów, natomiast tło – jeden z ośmiu kolorów. Dostępnych jest 256 różnych znaków, w tym wszystkie duże i małe litery alfabetu angielskiego, znaki przestankowe, niektóre litery alfabetów niemieckiego, greckiego i innych, a także symbole matematyczne oraz różne kształty, które mogą być pomocne, np. przy rysowaniu tabel. Ponieważ płytka zawiera 16 kB pamięci, to jednocześnie można zapamiętać cztery strony tekstu o szerokości wiersza 80 znaków. Dostęp do każdej ze stron nie jest związany ze stroną aktualnie wyświetlaną. Rozwiązanie takie umożliwia szybką zmianę wyświetlanego tekstu, bez konieczności zmiany zawartości pamięci.

Komputer może pracować w trzech różnych trybach graficznych. Pierwszy jest to grafika o małej rozdzielczości (100×160 punktów), przy czym każdy punkt może mieć jeden z 16 kolorów. W grafice o średniej rozdzielczości (320×200 punktów) mogą być wyświetlone jednocześnie tylko cztery kolory, przy czym tylko jeden z nich może być wybrany dowolnie, natomiast pozostałe trzy grupowo. W grafice o bardzo dużej rozdzielczości (640×200 punktów) są dostępne tylko dwa kolory, przy czym jeden z nich musi być kolorem czarnym. Jest to więc w zasadzie grafika monochromatyczna. Korzystając z trybu graficznego dysponuje się zestawem 128 podstawowych znaków. Istnieje możliwość zdefiniowania przez użytkownika dalszych 128 znaków.

Druga płytka, opracowana przez firmę IBM do monochromatycznego wyświetlania tekstów (bez możliwości graficznych), została w krótkim czasie wyparta przez wyrób firmy Hercules. Płytka ta umożliwia wydruk tekstu i obrazów graficznych. Rozdzielczość obrazu wynosi 720×348 punktów, a obraz jest znacznie lepszej jakości od tego, jaki uzyskuje się za pomocą płytki grafiki kolorowej.

KLAWIATURA

Klawiatura stanowi samodzielny człon IBM PC/XT/AT. Składa się z 83 klawiszy i jest połączona z komputerem przewodem o długości ok. 180 cm. Kąt jej ustawienia może być regulowany za pomocą nóżek. W części środkowej zamontowano zwykłą klawiaturę maszyny do pisania w układzie QWERTY. Po prawej stronie znajdują się powtórzone klawisze cyfr, klawisze sterujące ruchem kursora i inne, z lewej – dziesięć klawiszy funkcyjnych, których znaczenie można zmieniać programowo. Obsługą klawiatury i komunikacją z jednostką centralną zajmuje się mikroprocesor jednoukładowy Intel 8048.

Choć w chwili obecnej może się to wydać zaskakujące, klawiatura komputera IBM PC/XT była jednym z najczęściej krytykowanych elementów systemu. Podkreślano wprawdzie jej bardzo dobre właściwości mechaniczne, ale zwracano uwagę na niezbyt fortunne umiejscowienie niektórych klawiszy, takich jak *SHIFT*, *ALT* i *CAPS LOCK*, zbyt małą wielkość klawiszy *SHIFT* i *ENTER*, nieproporcjonalnie duży klawisz plusa i inne niedostatki. Nic więc dziwnego, że wiele firm zaczęło oferować „poprawione" klawiatury do tego komputera. Siła standardu jest jednak tak wielka, że w chwili obecnej układ klawiatury zaproponowany przez firmę IBM rozpowszechnił się nawet w wyrobach jej konkurentów.

ROZSZERZANIE KONFIGURACJI

Jedną z najważniejszych cech komputerów IBM PC/XT/AT jest łatwość dokonywania zmian ich sprzętowej konfiguracji. Na płycie głównej komputera znajduje się osiem gniazd, w których mogą być umieszczane dodatkowe płytki wzbogacające możliwości tego komputera. Zawierają one dodatkową pamięć, przetworniki A/C i C/A, dodatkowe łącza RS 232C, interfejsy do różnych standardów stosowanych w systemach przemysłowych i pomiarowych, grafikę o podwyższonej rozdzielczości itp. Istnieją już także płytki z mikroprocesorem Motorola 68000, umożliwiające uruchamianie programów działających pod nadzorem systemu operacyjnego Unix. Przyjęte przez konstruktorów rozwiązanie umożliwia wykorzystywanie komputerów IBM PC/XT/AT w wielu dziedzinach zastosowań.

OPROGRAMOWANIE

Oprogramowanie komputerów IBM PC/XT/AT, jako temat bardzo obszerny, będzie omawiane na łamach „Mikroklanu" w wielu innych artykułach. System operacyjny PC-DOS i jego kolejne wersje opracowała słynna firma Microsoft. Dostępne są także inne systemy operacyjne: CP/M-86, Xenix, Concurrent DOS (umożliwiający pracę w trybie wielozadaniowym), a także kompilatory większości popularnych języków programowania. Poza tym istnieje ogromne oprogramowanie użytkowe, szczególnie do prac administracyjno-biurowych, dla których istnieją gotowe rozwiązania, m.in. zintegrowane pakiety programowe (Lotus 1-2-3, Symphony). Do opracowywania tekstów służą m.in. programy WordStar (najpopularniejszy, choć wcale nienajlepszy), Professional Editor, EasyWriter. Bazy danych można zakładać korzystając z programu dBase II lub dBase III.

Komputery IBM PC/XT/AT są szeroko stosowane również do wspomagania prac projektowych. Można tu wymienić program AutoCAD (omawiany w bieżącym numerze „Mikroklanu"), wspomagający tworzenie rysunków architektonicznych, Smartwork – projektowanie płytek drukowanych, czy PCAD i RACAL-REDAC – tworzenie schematów układów logicznych, ich symulację i projektowanie płytek drukowanych.

KRZYSZTOF KONTEK

IBM PC/AT

Komputer IBM PC/AT został wprowadzony na rynek w trzy lata po premierze IBM PC. Projektując nowy wyrób, znacznie ulepszony w stosunku do pierwowzoru, uwzględniono doświadczenia zebrane z eksploatacji modelu PC. Poniżej zamieszczamy test przeprowadzony przez redakcję „Micro", który prezentuje wszechstronną ocenę walorów użytkowych modelu AT.

Model AT należy traktować jako odpowiedź IBM na zarzuty wysuwane w stosunku do wad i niedogodności występujących w poprzednich modelach PC. Oprócz tego zamierzeniem firmy było wprowadzenie możliwie najnowocześniejszych rozwiązań, które zapewniłyby temu potentatowi, jeśli nie czołową pozycję w świecie, to przynajmniej dobre miejsce w czołówce producentów komputerów osobistych.

POPRAWIONA KLAWIATURA

Jednym z najmocniej atakowanych elementów konstrukcji w modelu PC był układ klawiatury. Tak istotny dla operowania klawisz *RETURN* był trudno dostępny i przy szybkim palcowaniu stanowił źródło częstych błędów. W modelu AT nie można było usunąć wszystkich słabych punktów klawiatury, ponieważ na jej dotychczasowym układzie opierają się instrukcje użytkowania znacznej części oprogramowania. Przykładowo, tak jak poprzednio występują trzy różne klawisze *LOCK*, przełączające poszczególne części klawiatury na różne rodzaje funkcji. W wielu jednak szczegółach klawiatura została poprawiona, np. podzielono ją w bardzo wyraźny sposób na trzy pola.

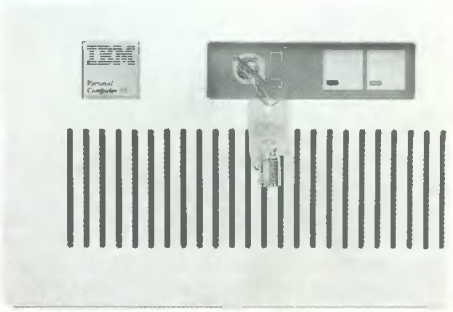
Niestety bez zmian pozostało dziesięć klawiszy funkcyjnych z lewej strony klawiatury. W polu głównym rzuca się w oczy powiększony klawisz *RETURN*, co znacznie ułatwia operowanie klawiaturą osobom używającym wszystkich dziesięciu palców. Większe i lepiej zlokalizowane są również oba klawisze przełączania na duże i małe litery. Niektóre klawisze, poprzednio z niezrozumiałych powodów przesunięte poza podstawową część klawiatury (np. klawisze nawiasów < >), umieszczono obecnie w tej części.

Również położony w prawo od głównego pola klawiatury łączny obszar kursora i cyfr został nieco uporządkowany. Cały układ jest bardziej klarowny, a podział na pola funkcyjne znacznie lepszy niż w wersji poprzedniej. Niestety nie została wyeliminowana podstawowa niedogodność, polegająca na umieszczeniu kursora wśród klawiszy cyfr.

Istotnym udogodnieniem są trzy diody świetlne znajdujące się powyżej pola cyfr, sygnalizujące operatorowi rodzaj trybu, w jakim w danym momencie działa klawiatura. Sprężystość klawiszy jest niemal idealna i pod tym względem klawiatura IBM ma wyraźną przewagę nad klawiaturami testowanego dotąd przez redakcję sprzętu kompatybilnego. W modelu AT zlokalizowano na czołowej stronie obudowy wyłącznik uruchamiany kluczykiem, co uniemożliwia korzystanie z klawiatury osobom nieupoważnionym, a jednocześnie nie powoduje przerwania przebiegającego w tym czasie procesu obliczeniowego. Kluczyk dodatkowo blokuje obudowę, uniemożliwiając jej zdjęcie. Szczegół ten jest bardzo istotny w sytuacji rosnącej przestępczości komputerowej.

ZMNIEJSZENIE GŁOŚNOŚCI

Należy stwierdzić, że model AT jest sprzętem skutecznie chroniącym system nerwowy użytkownika. Zarówno napędy dyskietek, jak dysku sztywnego, mają bardzo niski poziom głośności pracy. Wentylator chłodzący jest sterowany termostatem i nawet przy maksymalnym obciążeniu pracuje rzeczywiście cicho. Test wykazał, że głośność pracy AT nie przekracza 42 decybeli.



PROCESOR

Najistotniejszą innowacją w AT jest zastosowanie mikroprocesora 80286, który jest ok. 6 razy szybszy od swego poprzednika (8088) i teoretycznie może adresować pamięć o pojemności 16 MB. Jednak możliwości nie są w pełni wykorzystane przez nową wersję 3.0 systemu operacyjnego PC-DOS, który emuluje procesor 8088. Dlatego też pamięć, jaką mogą wykorzystywać programy, wynosi – tak jak poprzednio – maksymalnie 640 KB.

Dodatkową pamięć operacyjną, uzyskujemy poprzez dołączenie do systemu podprogramu, który pozwala symulować jedną lub więcej dyskietek jako pamięć RAM o maksymalnej pojemności 1 MB.

Rzeczywista szybkość wykonywania operacji przez AT jest wg opinii IBM trzykrotnie większe niż w modelu PC. Przeprowadzony test opinię tę w zasadzie potwierdził. W zastosowaniach, w których występują intensywne obliczenia, szybkość komputera można dodatkowo zwiększyć stosując procesor numeryczny 80287. Wymaga to oczywiście odpowiedniej modyfikacji dotychczasowego oprogramowania.

AT dysponuje również gniazdami do wprowadzenia dodatkowych kart. Dwa z nich przeznaczone są na karty stosowane już w modelu PC, a pięć dodatkowych umożliwia wykorzystanie całej szyny adresowej procesora 80286. Za ich pomocą pamięć AT można rozbudować do pojemności 3 MB.

DWIE WERSJE

Model AT jest obecnie produkowany w dwóch wersjach. Wersja 01 jest wyposażona w stację dyskietek o pojemności 1,2 MB oraz w pamięć operacyjną o pojemności 256 KB. Bardziej interesująca jest wersja 02, która dodatkowo dysponuje wbudowanym dyskiem sztywnym o pojemności 20 MB oraz pamięcią operacyjną 512 KB.

Z chwilą pojawienia się na rynku nowych rozwiązań technologicznych możliwe będzie rozszerzenie pojemności pamięci operacyjnej do 1 MB, bez konieczności stosowania dodatkowych kart.

Nowe stacje dyskowe o zwiększonej gęstości zapisu (ang. high density) typu YE-Data 380 są przeznaczone dla dyskietek specjalnych, które poza IBM produkują obecnie tylko firmy BASF oraz Maxell. Stacje te pozwalają również odczytywać dyskietki zapisane zgodnie ze starym formatem PC oraz (co jednak nie zostało oficjalnie potwierdzone) zapisywać je i formatować. W wersji 02 można dodatkowo przyłączyć drugą stację dyskietek, zarówno konwencjonalną, jak i o zwiększonej gęstości zapisu.

Dysk sztywny o pojemności 20 MB jest nieco szybszy od stosowanego w modelu XT dysku 10 MB. Zamiast drugiej stacji dyskietek można przyłączyć dodatkowy dysk sztywny, co sprawia, że AT uzyskuje pamięć zewnętrzną o imponującej łącznej pojemności 41,2 MB. Należy spodziewać się, że producenci pamięci dyskowych w niedługim czasie pojemność tę jeszcze powiększą.

W konfiguracji podstawowej model AT zawiera wbudowany zegar z kalendarzem, który po włączeniu komputera inicjuje automatycznie działanie systemu operacyjnego. Ładowana podczas eksploatacji komputera bateria zasilająca zegar eliminuje potrzebę jego każdorazowego ustawiania. Zasila ona również specjalną pamięć, zawierającą informację na temat konfiguracji AT. Umożliwia to określanie aktualnej konfiguracji za pomocą programów instalacyjnych.

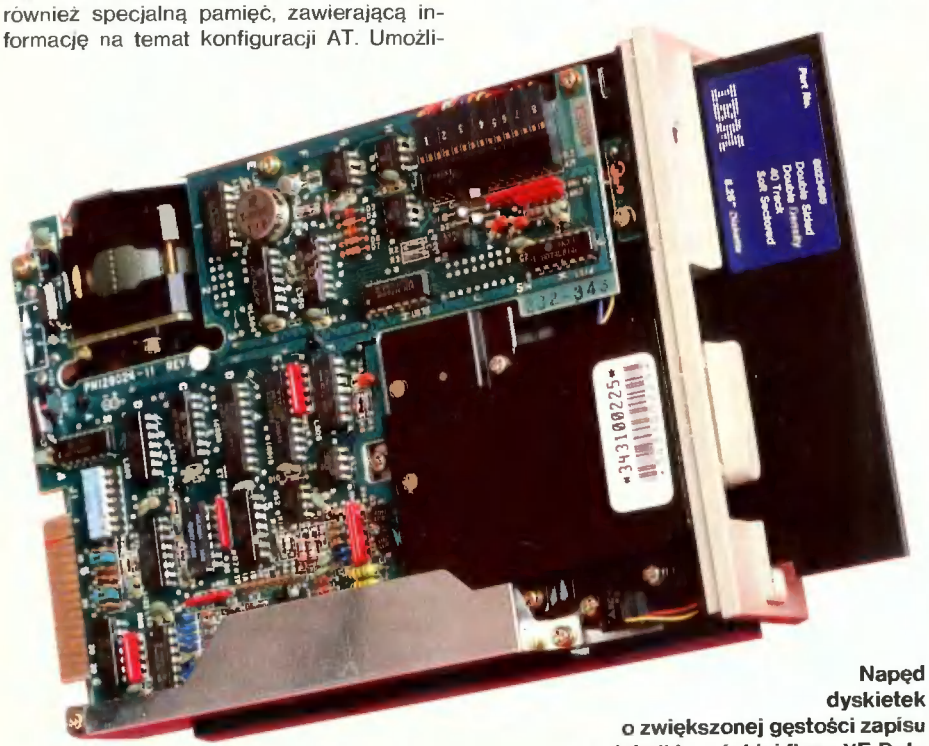
PC-DOS 3.0

Jak już wspomniano, dla modelu AT opracowano nową wersję systemu operacyjnego PC-DOS. Wersja 3.0 tego systemu, oprócz podprogramów dla dysku 20 MB oraz nowych dyskietek, zawiera kilka dodatkowych instrukcji. Instrukcja *ATTRIB*, chociaż nie jest to osiągnięcie rewolucyjne, powoduje blokadę zapisu w zbiorze danych (tylko możliwość odczytu). Instrukcja *LABEL* umożliwia wprowadzenie lub zmianę nazwy dyskietki lub dysku, również po operacji formatowania.

Bardziej interesująca jest instrukcja *SHARF*, umożliwiająca wspólne wykorzystywanie dysków sztywnych w ramach sieci komputerowej. W dokumentacji wymieniono również instrukcję *LASTDRIVE*, która nadaje nazwę ostatnio wykorzystywanemu przez system operacyjny napędowi dysku. Podczas próby z dostarczoną do testowania dyskietką systemową wprowadzenie tej instrukcji spowodowało jednak komunikat „Bad command or filename" (błędna instrukcja lub nazwa zbioru). Poza tym niejasny jest głębszy sens *LASTDRIVE*.

Nie licząc tej usterki stwierdzono, że PC-DOS 3.0 działa poprawnie również na dotychczasowych modelach PC, a także na komputerach kompatybilnych. System ten również automatycznie identyfikuje każdorazową konfigurację sprzętu. Wersja 3.0 w porównaniu z poprzednią wersją 2.1 zajmuje tylko ok. 10 KB więcej komórek pamięci.

AT, podobnie jak PC, dysponuje również językiem BASIC zawartym w pamięci ROM. Język ten został rozszerzony o kilka nowych instrukcji. Instrukcja *SHELL* umożliwia wykonanie innego programu z poziomu języka BASIC, jednakże podczas testu stwierdzono, że działanie to często prowadzi do załamania systemu. Za pomocą instrukcji *ENVIRON* można określić oraz zmodyfikować niektóre parametry systemu. Pozostaje jednak generalna wątpliwość, jakie uzasadnienie ma interpreter BASICA w tak kosztownym sprzęcie profesjonalnym.



Napęd dyskietek o zwiększonej gęstości zapisu produkcji japońskiej firmy YE-Data

XENIX

Oprócz systemu PC-DOS, model AT wyposażono w wariant systemu operacyjnego Unix, który firma Microsoft nazwała Xenix. Pozwala on bezpośrednio korzystać z całej pojemności pamięci operacyjnej komputera, umożliwia także eksploatację, wraz z AT, trzech dodatkowych komputerowych stanowisk pracy. IBM traktuje Xenix głównie jako narzędzie do realizacji zastosowań specjalistycznych, np. na wyższych uczelniach. W tej sytuacji Unix wśród systemów operacyjnych mikrokomputerów będzie odgrywać w najbliższym czasie jeszcze stosunkowo niewielką rolę.

Przeszkodą, jaką w ostatnich latach musiało przezwyciężyć wiele mikrokomputerów, było osiągnięcie kompatybilności z IBM PC. Obecnie sam IBM znalazł się w podobnym położeniu, ponieważ wymaganie takie musi spełnić sam. Dotychczasowe wyniki są jednak gorsze niż w wielu naśladownictwach PC. IBM przyznaje, że większości dotychczasowych kart różnych producentów nie można zastosować w modelu AT (np. karta rozszerzająca pamięć, interfejs Centronics do drukarki).

Nieco lepiej przedstawia się sytuacja w dziedzinie oprogramowania użytkowego. Lotus 1-2-3, WordStar czy Multiplan działają w modelu AT bez jakichkolwiek trudności, tak samo jak TURBO-Pascal i większość kompilatorów firmy Microsoft. Niepowodzeniem kończą się natomiast próby wykorzystania starszych wersji BASICA. Również klasyczny test kompatybilności, jakim jest Flight Simulator firmy Microsoft nie „wystartował" na AT. Ze źródeł amerykańskich potwierdza się wiadomość, że dotyczy to również programów EasyWriter, pfs: File, pfs: Report oraz Visicalc. Dlatego każdemu, kto zamierza nabyć AT a dysponuje już oprogramowaniem PC, należy doradzić, aby przed ostateczną decyzją zakupu oprogramowanie to przetestował u sprzedawcy.

* * *

IBM AT jest bardzo szybkim i w wielu szczegółach poprawionym modelem PC. Poza wspomnianymi przykładami niekompatybilności użytkownik nie natrafi na szczególne problemy eksploatacyjne. Najbardziej odczuwalne zalety AT to cicha praca oraz znacznie poprawiona klawiatura. Czy tego rodzaju zalety uzasadniają wyższą cenę tego modelu, zależy oczywiście od rodzaju planowanego zastosowania. Nabywając AT, uzyskać można w każdym razie urządzenie dające pewność, że nawet w ciągu kilku lat nie stanie się ono sprzętem przestarzałym.

micro

mikroklan

AutoCAD

Dotychczas pakiety programowe CAD stosowano jedynie na dużych komputerach lub przynajmniej na minikomputerach. Ostatnio sytuacja uległa zmianie. Liczne dziedziny, dotąd zarezerwowane wyłącznie dla dużych i średnich maszyn, zostały „odkryte" dla mikrokomputerów. Między innymi projektowanie wspomagane komputerem – CAD, a także wytwarzanie wspomagane komputerem – CAM (Computer Aided Manufactura). Przykładem systemu CAD może być pakiet AutoCAD stosowany m.in. na komputerach osobistych IBM i kompatybilnych. Jest to dwuwymiarowy system graficzny, umożliwiający tworzenie za pomocą mikrokomputera wszelkiego rodzaju rysunków, planów czy projektów.

AutoCAD może znaleźć zastosowanie przede wszystkim w biurach projektowych zatrudniających architektów, geodetów i konstruktorów. Można go porównać z systemem przetwarzania tekstów, tyiko że zamiast tekstów powstają tu najbardziej złożone rysunki.

SPRZĘT

Podręcznik informuje, że do stosowania pakietu potrzeba przynajmniej 364 KB pamięci. W praktyce okazuje się, że system może być używany już przy pojemności 256 KB. Istnieje jednak pewne ograniczenie: jeśli wprowadzi się duży skomplikowany rysunek, może się zdarzyć, że bufory przeznaczone na stos i dane będą ze sobą kolidować. W takim wypadku AutoCAD podaje komunikat *AutoCAD gives up* (AutoCAD poddaje się) i następuje przejście do systemu operacyjnego, którym jest MS-DOS. Standardowa konfiguracja mikrokomputera VICTOR, pracującego pod nadzorem systemu operacyjnego MS-DOS, wystarczy do współpracy z programem AutoCAD za pomocą klawiatury. Jeśli zamierza się stosować wygodniejsze urządzenia zewnętrzne, na przykład pióra świetlne (Sun-Flex Touch Pen), trzeba zainstalować dodatkową płytę sterującą oraz ekran dotykowy.

Najlepiej jednak używać pulpitu digitajzera z myszką. Za ich pomocą można najszybciej wybierać potrzebne cechy rysunku. Program może pracować z jednym spośród 8 różnych pulpitów i jednym spośród 5 różnych ploterów. Rysunek nie musi być wyprowadzany na ploterze. W celu otrzymania prostszej, lecz za to szybkiej kopii, może być on wyprowadzany na drukarce z możliwością zarówno poziomego, jak i pionowego ustawienia. Dodatkową opcją sprzętową jest zastosowanie zmiennoprzecinkowego koprocesora arytmetycznego, znacznie skracającego czas obliczeń. Z pewnością warto mieć możliwie dużą pojemność pamięci RAM. AutoCAD stara się przechowywać rysunek w pamięci RAM, aby zredukować do minimum czasochłonny dostęp do dyskietek. Znaczne zwiększenie wydajności systemu można uzyskać przez zastosowanie sztywnego dysku.

ZASADA DZIAŁANIA

Program działa częściowo w trybie konwersacyjnym, a częściowo w trybie menu. Po zainicjowaniu programu użytkownik znajduje się w menu głównym, gdzie może wybrać następujące opcje: tworzenie nowego rysunku, zmiany w starym rysunku, konfigurowanie AutoCAD, listowanie zbiorów i

wyprowadzanie rysunku. Po dokonaniu wyboru użytkownik przechodzi do menu podrzędnego. Jeśli wybrał tworzenie lub zmiany rysunku, to znajduje się w edytorze graficznym stanowiącym jądro pakietu. Ekran w edytorze jest podzielony na trzy części: największa – przeznaczona jest na rysunek, pod nią są trzy wiersze zarezerwowane na wyświetlanie komunikatów, zaś z prawej strony widnieje lista rozkazów. Dodatkowo, za pomocą klawisza funkcyjnego *FLIP-SCREEN* można wybierać między ekranem z grafiką a ekranem z tekstem, który zawiera ostatnio wprowadzone 24 wiersze tekstu. Mikrokomputer musi więc w każdej chwili przechowywać w pamięci RAM dwa kompletne obrazy, co zajmuje 80 KB. Treść trzech wierszy tekstu na ekranie z grafiką jest identyczna z treścią trzech ostatnich wierszy ekranu z tekstem. Rozkazy mogą być wprowadzane dwoma sposobami. Pierwszy – to zwykłe wystukanie rozkazu na klawiaturze, zakończone wciśnięciem klawisza *RETURN*. Drugi sposób polega na użyciu myszki. Przy jej pomocy naprowadza się kursor na ekranie na odpowiednią pozycję listy rozkazów i wciska umieszczony na myszce klawisz.

Dobrym pomysłem jest bardzo pomocny w pracy tryb powtarzania. Wciśnięcie samego klawisza *RETURN* lub klawisza spacji powoduje powtórne wykonanie ostatniego rozkazu, niezależnie od tego czy był on wprowadzony z klawiatury, czy też za pomocą myszki.

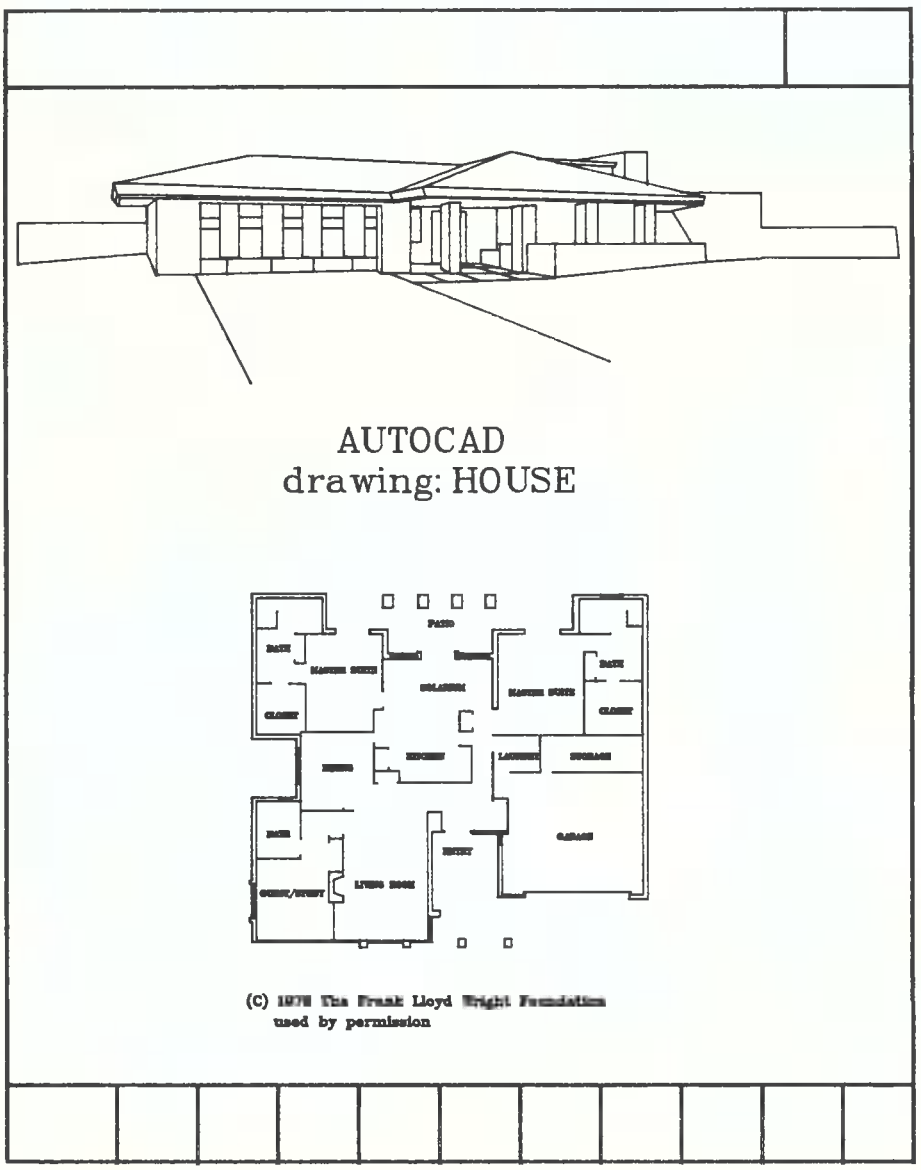
Wiele rozkazów używanych w trybie powtarzania pozwala na skrócone wprowadzanie danych. Kursor używany przez edytor ma kształt krzyża wielkości 25 punktów lub zajmującego cały ekran. Kursor może być ustawiany za pomocą myszki lub klawisza kursora znajdującego się na klawiaturze. Szybkość przesuwania się kursora można zmieniać klawiszem funkcyjnym.

Do lokalizacji punktów na rysunku używana jest siatka współrzędnych. Każdemu punktowi rysunku przyporządkowana jest para współrzędnych w układzie *XY*.

Na przykład, odcinek można wprowadzić podając współrzędne punktów końcowych. Jednostką jest jednostka miary używana do opisu całego rysunku, np. cal, stopa, centymetr, angström. Przy wyprowadzaniu na ploter można stosować współczynnik skalujący, który pozwoli na uzyskanie rysunku o potrzebnych wymiarach.

Dane wewnątrz komputera są przechowywane w postaci zmiennoprzecinkowej. Stosunek wielkości największych zapamiętanych obiektów do najmniejszych może być co najmniej 1 000 000 : 1. Rysunek jest generowany za pomocą następujących rozkazów podstawowych: linia, pas, okrąg, łuk, punkt, powierzchnia, tekst, seria obiektów, kształty i bloki. Oprócz rozkazu podstawowego podaje się zwykle dodatkowe informacje. Na przykład, można podać punkt, w którym zostanie umieszczony określony element. Inne rozkazy wymagają podania wartości liczbowych, określających wysokość, szerokość lub kąt. AutoCAD automatycznie żąda podawania odpowiednich informacji.

Różne pytania i wymagane odpowiedzi są dokładnie opisane i zilustrowane przykładami w podręczniku. Często na jedno pytanie można udzielić wielu różnych odpowiedzi. Przykładowo, polecenie wprowadzenia punktu można wykonać za pomocą kursora w kształcie krzyża lub podając współrzędne z klawiatury. Współrzędne można podawać jako bezwzględne lub względne. Dwa przykłady wprowadzania względnego: *§2, –1* oznacza wprowadzenie nowego punktu: dwie jednostki w prawo i jedną jednostkę w dół, natomiast *§2 < 45* – wprowadzenie nowego punktu oddalonego o dwie jednostki pod kątem 45 stopni od poprzedniego punktu. Odległości lub wektory przesunięcia wprowadzane są na ogół przez podanie dwóch punktów lub wektora przesunięcia.



ROZKAZY

Najprostszym rozkazem jest rozkaz wprowadzania odcinka prostej – LINE. Po jego wprowadzeniu system natychmiast pyta: *od jakiego punktu?* Teraz wybierany jest punkt początkowy. Po wprowadzeniu pojawia się on na ekranie w postaci małego krzyżyka. Następnie system pyta: *do jakiego punktu?* Przesuwany kursor będzie ciągnął za sobą odcinek wychodzący z punktu początkowego. Dzięki temu zawsze widać, jak umiejscowiony został wprowadzany odcinek. Czasami trzeba wprowadzić kilka odcinków zależnych od siebie. Kolejny odcinek może zaczynać się w końcowym punkcie poprzedniego odcinka. Jeśli na pytanie o punkt koncowy wprowadzimy C (ang. close – sąsiedni), koniec rysowanej linii będzie jednocześnie początkiem następnej.

Okrąg rysuje się za pomocą rozkazu *CIRCLE*, wprowadzając punkt środkowy i promień lub też trzy punkty leżące na tym okręgu. ARC służy do rysowania łuków czyli fragmentów okręgu. Aby pokazać różnorodność sposobów wprowadzania danych oferowaną przez AutoCAD, przytoczymy listę możliwości definiowania łuków: 1) – trzy punkty, 2) – punkt początkowy, środkowy i kąt, 3) – punkt początkowy, środkowy, cięciwa, 4) – punkt początkowy, końcowy, promień, 5) – punkt początkowy, końcowy, kąt, 6) – przedłużenie wcześniejszej linii lub łuku. Często linie muszą mieć określoną grubość. Można je wtedy rysować za pomocą rozkazu

TRACE (ang. ślad). Ślady takie są wprowadzane tak jak linie, tyle że AutoCAD pyta wpierw o szerokość pasa.

W odróżnieniu od wielu innych programów graficznych operowanie tekstami w AutoCAD jest bardzo wygodne. Za pomocą rozkazu *TEXT* można dowolny tekst umieścić na rysunku z dosunięciem w lewo lub w prawo. Co więcej, istnieje możliwość ustawienia tekstu w wydzielonym obszarze. Jeśli chcemy dopasować tekst do określonego pola, wystarczy wprowadzić tylko punkt początkowy i końcowy, AutoCAD zaś wybierze automatycznie wielkosc liter. Można też zmieniać w szerokim zakresie wysokość liter oraz kąt ich nachylenia. Jeśli potrzebny jest nowy zestaw znaków, to wystarczy go wprowadzic rozkazem *LOAD*. Do dyspozycji są trzy zestawy znaków. Można też tworzyć własne zbiory znakow. Następne rozkazy są bardziej skomplikowane i do ich realizacji trzeba dokonać wyboru grupy elementów. AutoCAD żąda w tym celu: *Wybierz obiekty lub okienko lub figurę poprzednią.*

Wybór obiektu można przeprowadzić trzema sposobami. Można wprowadzać kolejne punkty znajdujące się jak najbliżej poszukiwanego obiektu. AutoCAD podaje wtedy w odpowiedzi liczbę znalezionych obiektów, jakie odpowiadają kryterium wyboru. Druga możliwość to określenie okienka. AutoCAD pyta w tym wypadku o lewy dolny, a po jego wprowadzeniu – o prawy górny róg okienka, które ukazuje się na ekranie. Za pomocą myszki można przesuwać pozycję kursora pokrywającą się z prawym górnym narożnikiem okienka i tym samym zmieniać jego położenie. Także w tym wypadku AutoCAD informuje, ile znaleziono obiektów.

Obiekt zostanie uznany za znaleziony, jeśli każda należąca do niego część znajdzie się w wybranym okienku. Ma to znaczenie, na przykład, przy wymazywaniu części rysunku za pomocą rozkazu *ERASE*. Najlepiej zrealizować rozkaz *ERASE* za pomocą trzeciej metody – przez podanie litery *L* (skrót od ang. last – ostatni). Zmazywany jest wtedy ostatnio wprowadzony obiekt. W praktyce okazuje się, że rozkaz *ERASE* jest często używany zbyt pochopnie i powoduje zmazanie obiektów, które jeszcze są potrzebne. Pomocą jest w tym wypadku rozkaz *OOPS*. Przywraca on obiekty zmazane ostatnim rozkazem *ERASE*. Obiekty zmazane wcześniejszymi rozkazami *ERASE* nie są już zapamiętywane i dlatego nie mogą być odtworzone.

ROZKAZY EDYCJI

Przedstawimy teraz rozkazy służące do manipulowania fragmentami rysunku. Przypuśćmy, że po wprowadzeniu fragmentu rysunku okazało się, że należy go umieścić wyżej, aby zostało pod nim jeszcze nieco miejsca na kilka linijek tekstu. Problem można rozwiązać przy użyciu rozkazu *MOVE*. Wystarczy nakierować pojawiające się po wprowadzeniu rozkazu *MOVE* okienko celownicze na obiekt, który chcemy przesunąć. AutoCAD żąda następnie wprowadzenia wektora przesunięcia, w sposób względny z klawiatury bądź graficznie za pomocą kursora.

Podobnie działa rozkaz COPY. W tym wypadku wybrany obiekt zostanie powielony i wprowadzony na rysunku w miejscu określonym przez wektor przesunięcia.

Bardzo przydatny jest rozkaz *ARRAY*, który pozwala zaoszczędzić wiele czasu. Przy jego użyciu można wielokrotnie kopiować wybrane obiekty, rozmieszczając je w układzie prostokąta lub okręgu. Każdy z tak powstałych obiektów może być potem indywidualnie przesuwany i transformowany. Rozkaz ten można stosować nawet bez zaglądania do podręcznika. Dialog systemu AutoCAD z użytkownikiem jest tak klarowny, że niepotrzebne są dodatkowe wskazowki. Odnosi się to właściwie do wszystkich rozkazów systemu. Typowym zastosowaniem rozkazu *ARRAY* jest rysowanie elewacji domu. Po narysowaniu jednego okna wystarczy podać współrzędne pozostałych – zostaną one automatycznie skopiowane.

Innym przydatnym dla architektów rozkazem jest *AREA*. Przy jego pomocy można obliczać pole powierzchni dowolnego wieloboku, wprowadzając punkty jego wierzchołków.

WPROWADZANIE RYSUNKÓW

AutoCAD pozwala na tworzenie biblioteki fragmentów rysunków. Przy tworzeniu różnorodnych rysunków, fragmenty te są pobierane z biblioteki i umieszczane w dowolnym miejscu rysunku. Fragmentem zapamiętanym w bibliotece może być dowolna część dowolnego rysunku. Do zapamiętania fragmentu rysunku służy instrukcja *BLOCK*, zaś do jego pobierania instrukcja *INSERT*, przy czym możliwa jest tu zmiana skali rysunku (powiększenie lub zmniejszenie).

POMOCE RYSUNKOWE

Niezwykle użyteczny jest rozkaz *SNAP*. Wprowadzane punkty są rozmieszczane w siatce współrzędnych o wybieranej rozdzielczości, która definiuje odstęp pomiędzy punktami. Współrzędne kursora, a co za tym idzie także i wszystkie wprowadzane współrzędne, są zaokrąglane w taki sposób, aby zdefiniowany przez nie punkt był tożsamy z najbliżej położonym punktem siatki, jeśli tylko tryb *SNAP* został uaktywniony. Rozdzielczość *SNAP* można w każdej chwili zmienić.

Trochę podobną funkcję spełnia rozkaz *GRID* (raster). *GRID* powoduje pokrycie rysunku siatką punktów o dowolnie wybranym odstępie pomiędzy punktami. Pozwala to lepiej orientować się w wielkosci elementów i ich wzajemnej zależności. Siatkę (raster), podobnie jak *SNAP*, można w każdej chwili włączyć lub wyłączyć, gdyż nie stanowi ona części składowej rysunku.

Do rysowania wyłącznie poziomych i pionowych linii służy rozkaz *ORTHO*. Jeśli stan *ORTHO* jest aktywny, to wszystkie linie czy pasy będą rysowane jako ortogonalne do układu współrzędnych. Bardzo użyteczny jest rozkaz *LAYER* (warstwa). W rysunku można wyróżnić 127 warstw. Rozkaz *LAYER* pozwala grupować fragmenty rysunku na jednej z tych warstw. Przy tym kolory warstw, jak również ich kombinacje mogą być zmieniane. Najprościej wyobrazić sobie można warstwy jako przezroczyste folie z rysunkami, które można dowolnie umieszczać jedna na drugiej oraz zabierać i dodawać. Przykładowo, przy projektowaniu rozłożenia elementów na płytce drukowanej można wyróżnić pięć następujących warstw:

- warstwa 1, czerwona – druk na stronie elementów
- warstwa 2, biała – otwory i ścieżki zasilania
- warstwa 3, niebieska – druk na stronie lutowania
- warstwa 4, żółta – opis wykonany sitodrukiem
- warstwa 5 – opcjonalne wymiarowanie.

Można wybrać na przykład tylko takie przedstawienie, na którym widoczne są ścieżki zasilania i druk na stronie lutowania, lub dowolną inną kombinację. AutoCAD wykreśla tylko widoczne warstwy. Dzięki możliwosciom zmiany kolorów poszczególnych warstw wielobarwne rysunki można drukować w formie jednokolorowej na jednobarwnych ploterach. Dużą wygodę stanowi włączenie i wyłączanie trybów *SNAP*, *GRID* i *ORTHO* za pomocą klawiszy kontrolnych (np. jednoczesne wciśnięcie klawisza CTRL oraz klawisza S włącza tryb *SNAP*).

EKRAN

Pierwszy wiersz ekranu jest używany do wyświetlania statusu. Informuje on, które tryby pomocnicze są w danej chwili aktywne. Dodatkowo wyświetlane są wartości liczbowe współrzędnych, które zmieniają się wraz z ruchem kursora po ekranie.

Często używanym rozkazem jest *ZOOM*. Jeśli określony fragment rysunku chcemy narysować szczególnie dokładnie, to wybrany wycinek jest powiększany przez wybór odpowiedniego okienka. Sam rysunek w pamięci nie ulegnie zmianie, jedynie wybrany fragment zostanie powiększony do rozmiarów całego ekranu. Powiększone wycinki rysunku można przesuwać przez podanie rozkazu *PAN* i wektora przesunięcia.

WYSPECJALIZOWANE POMOCE KREŚLARSKIE

Tak można nazwać pakiet rozszerzający AutoCAD. Zawiera on wiele dodatkowych rozkazów. Dodatkową pomocą w kreśleniu jest na przykład opis osi współrzędnych. Rozkaz *BREAK* służy do wymazania części linii, pasa, okręgu lub łuku. Można też przeprowadzić automatyczne zaokrąglanie kątów.

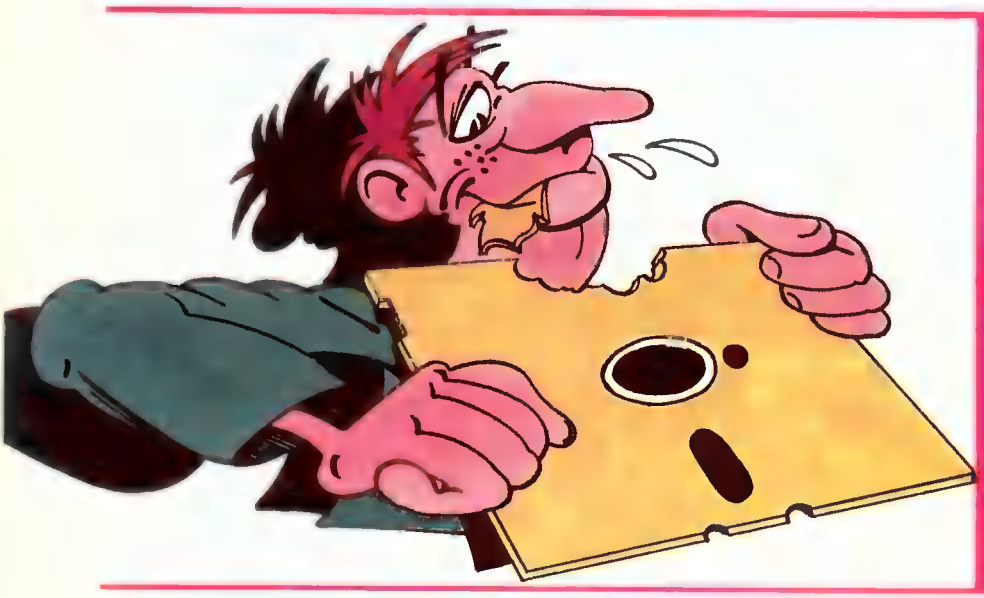
FILLET łączy dwie linie za pomocą łuku okręgu o zmiennym promieniu. Dużą pomocą jest półautomatyczne wymiarowanie obiektów. Użytkownik musi jedynie określić położenie linii wymiarowych – strzałki, liczby itp. będą narysowane przez system AutoCAD. Szesnaście stron podręcznika poświęcono kreskowaniu powierzchni. Do dyspozycji jest 41 wzorów kreskowania. Jeśli to dla kogos za mało, to może sam zdefiniować dodatkowe wzorce i zapamiętać je na dysku.

Oprócz opisanych tu zalet AutoCAD ma też pewne wady, nie są one jednak zbyt istotne i dlatego nie będziemy ich tu dokładnie omawiać. W porównaniu z innymi systemami o podobnej wydajności jego cena (poniżej 200 dol. za oba pakiety) wydaje się być stosunkowo niska.

micro

Jakość dyskietek (1)

Dyskietka jako nośnik danych zyskała obecnie dominującą pozycję w eksploatacji mikrokomputerów. Dla użytkownika staje się ona po pewnym czasie typowym materiałem eksploatacyjnym, wybieranym najczęściej według kryterium ceny. Na temat jakości tego nośnika nie można praktycznie uzyskać rzeczowych informacji, ponieważ dostępne publikacje producentów mają ze zrozumiałych względów charakter reklamowy. W dwóch kolejnych artykułach przedstawione zostaną wyniki badań dyskietek różnych producentów. Część pierwsza dotyczy ich własności mechanicznych.



Chociaż spadek cen oraz znacznie większa pojemność wzmogły atrakcyjność dysku sztywnego (ang. hard disk), dyskietka utrzymuje nadal dominującą pozycję jako nośnik szczególnie wygodny do przekazywania i przechowywania danych. 134 mln dyskietek sprzedanych w 1984 r. tylko na terenie Europy, najlepiej świadczy o masowości stosowania tego nosnika. Przeważającą część (55%) stanowiły minidyskietki o średnicy 5 1/4 cala (130 mm). Ponieważ mikrodyskietki o średnicy 3 1/2 cala (90 mm) przy mniejszych wymiarach mają taką samą, a często nawet większą pojemność, należy spodziewać się, że prędzej czy później właśnie one zdobędą przeważającą część rynku. Wyniki badan amerykanskich stwierdzają, że nastąpi to juz w 1989 roku.

Dyskietki standardowe o średnicy 8 cali (200 mm) oraz wspomniane już minidyskietki mają identyczną konstrukcję. Składają się one z następujących elementów:

- krążka folii magnetycznej (ang. cookie)
- obwoluty ochronnej (ang. jacket)
- otuliny (ang. liner).

Folia magnetyczna składa się z poliestrowego podłoża grubości 75 µm, pokrytego z obu stron warstwą tlenku żelaza grubości kilku µm. Obwoluta ochronna jest wykonana najczęściej z polichlorku winylu oraz wyłożona otuliną z bezwłóknistego tworzywa. Ma to zapobiec uszkodzeniom warstwy magnetycznej podczas operowania dyskietką oraz nie dopuszczać do niej kurzu.

BUDOWA DYSKIETEK

Dyskietka to krążek folii magnetycznej z wyciętymi dwoma otworami. Większy, umieszczony pośrodku, ma średnicę 38,1 lub 28,57 mm i służy do mocowania dyskietki na wrzecionie napędu. Odchylenie normatywnego położenia tego otworu, gwarantujące zachowanie niezbędnej dokładności rozmieszczenia ścieżek nie może przekroczyć 25 µm. W pobliżu otworu środkowego znajduje się mniejszy otwor zwany indeksowym. Służy on do określenia pozycji dyskietki podczas jej ruchu obrotowego w napędzie. W obwolucie ochronnej, oprócz otworów środkowego oraz indeksowego, wycięty jest także otwor umożliwiający kontakt warstwy magnetycznej z głowicą zapisu-odczytu. W dyskietce o średnicy 130 mm do zapisu wykorzystywany jest jedynie obszar zawarty pomiędzy okręgami o promieniach 34,0 i 57,4 mm. Odpowiednio do tego otwór (szczelina) zapisu-odczytu ma wymiar 35×6 mm.

Prostokątna szczelina z prawej strony szczeliny zapisu-odczytu służy do blokady zapisu. Jeśli jest zakryta, dyskietka nie może być zapisywana. W dyskietkach o średnicy 200 mm blokada jest zrealizowana odwrotnie: zabezpieczenie przed skasowaniem wymaga odkrycia szczeliny. Odpowiednie kawałki folii do zakrywania szczeliny blokady zapisu producenci wkładają do opakowań wraz z samoprzylepnymi etykietami, przeznaczonymi do opisania zawartosci dyskietki.

Małe okrągłe nacięcia znajdujące się na dolnej stronie dyskietki zapobiegają naprężeniom mechanicznym, jakie występują wskutek gięcia, np. gdy dyskietkę wsuwa się nierówno do napędu.

Dyskietka kierowana do sprzedaży jest pakowana w dodatkową osłonę, ktora ma zapobiec uszkodzeniu warstwy magnetycznej podczas transportu i składowania. Do tego celu używa się tradycyjnego kartonu lub tworzyw, przypominającego cienki papier woskowy. Ma ono szereg cennych zalet, takich jak odporność na strzępienie, przerwanie oraz absorbcję zanieczyszczeń i wody. Osłony z niego w porównaniu z kartonem mają jednak tę wadę, ze nie chronią dyskietki przed obciążeniami mechanicznymi, a więc powodując konieczność uważniejszej eksploatacji.

Po zakończeniu procesu wytwórczego każda dyskietka wymaga określenia jej cech jakościowych. Polega to na wyborze sposobu zapisu (jedno- lub dwustronny) oraz gęstości rozmieszczenia ścieżek. Podjęcie odpowiedniej decyzji opiera się na bardzo dokładnej kontroli magnetycznej tych części powierzchni dyskietki, które będą służyły do zapisu i odczytu danych. Najbardziej renomowani producenci przeprowadzają kontrolę każdej ścieżki, inni sprawdzają tylko ścieżki określone normą, wreszcie inni – całą powierzchnię zapisu, ■ więc również obszar między ścieżkami. Ta ostatnia metoda jest najbardziej skuteczna, ponieważ wskutek odchyleń w ustawieniu głowic oraz wpływu temperatury, pojawić się może ewentualność zejścia z właściwego toru ścieżki. Kontrola całej powierzchni pozwala więc zapisywać bez obaw również w obszarze międzyścieżkowym.

Kontrola warstwy magnetycznej polega na wykryciu brakujących sygnałów zapisu oraz sygnałów, jakie pozostały po jego skasowaniu. Dyskietka właściwej jakości nie powinna wykazywać żadnej z wymienionych usterek.

WŁASNOŚCI MECHANICZNE

Zachowanie się dyskietki podczas jej ruchu wirowego w napędzie oraz synchronizacja oddziaływania głowicy na powierzchnię magnetyczną są problemami tak złożonymi, że na ich temat nie można sformułować ogólnie obowiązujących reguł. Wymagania stawiane dyskietkom są w znacznym stopniu zależne od konstrukcji napędu. Dotyczy to zwłaszcza centrycznego mocowania dyskietki oraz jej kontaktu z głowicą.

Napędy przystosowane do zapisu jedno- i dwustronnego opierają się na odmiennych zasadach działania. Jednostronne mają tylko jedną głowicę, która zapisuje na odwrotnej stronie dyskietki zwanej stroną 0 (stronę czołową, zwaną stroną 1, rozpoznaje się po nakiejonej etykiecie firmowej). Docisk głowicy do powierzchni dyskietki jest w tej konstrukcji bardziej intensywny niż w napędach dwustronnych, gdzie dwie głowice są położone naprzeciw siebie. Rozwiązanie to wymaga użycia znacznie mniejszych sił dociskowych. Ze względu na większą pojemność ostatnio coraz powszechniej są stosowane napędy z zapisem dwustronnym. W uzupełnieniu tych informacji należy wymienić następujące praktyczne zalecenia:

■ dyskietki zapisywane dwustronnie powinny być lepszej jakości,

■ często praktykowane przez mniej doświadczonych użytkowników odwracanie dyskietek w napędach z zapisem jednostronnym jest niewskazane.

W odróżnieniu od dyskietek, głowice są zbudowane z bardzo twardych materiałów i praktycznie nie ulegają zużyciu. Istotne jest więc utrzymywanie głowicy w czystości, ponieważ pył powstający ze ścierania powierzchni magnetycznej dyskietek może utworzyć na niej rodzaj skorupy, powodującej powstawanie błędów. Czyszczenie głowic odbywa się za pomocą specjalnych dyskietek, w których warstwę magnetyczną zastąpiono warstwą czyszczącą. Może to odbywać się na sucho lub z nawilżaniem alkoholem. Firma Maxell zaleca np. metodę mokrą, która jej zdaniem całkowicie eliminuje niebezpieczeństwo uszkodzenia głowicy. Istnieje znaczna rozbieżność poglądów na temat częstotliwości czyszczenia głowic. Autor na podstawie własnych doświadczeń zaleca okresy tygodniowe, jeżeli napęd działa więcej niż jedną godzinę dziennie.

MECHANICZNE WŁASNOŚCI DYSKIETEK

Producent	Symbol wyrobu	Grubość osłony (mm)	Pierścień usztywniający	Połysk	Szum wirowania
BASF	FDS 73041	0,20	jest	3	średni
D&B	Disky 3222	0,24	jest	5	mały
3M	746-0	0,27	jest	3	mały
Dysan	204/2D	0,22	jest	1	duży
Elephant	No. ■	0,22	jest	2	duży
Fuji	MD 2D 96	0,20	jest	2	mały
Maxell	MD 2-DD	0,21	brak	1	średni
Memorex	1D-80	0,23	jest	2	mały
RPS	MN 1-DD	0,22	jest	3	mały
TDK	M2 DX-5	0,22	brak	3	mały
Verbatim	577-01	0,21	jest	1	mały
Xidex	5022-2000	0,25	jest	4	średni

ŻYWOTNOŚĆ DYSKIETEK

Żywotność dyskietek określają producenci liczbą od 4 do 70 mld dostępów głowicy do ścieżek. Napędy dyskietek 5 1/4 cala działają zwykle z szybkością 300 obr./min. Jeżeli więc przez dłuższy czas jest zapisywana ta sama ścieżka, to okres użytkowania dyskietki wyniesie 200-3800 godzin pracy. Wielkość i rozpiętość tych liczb wyjaśniają, dlaczego do programu badań nie włączono testu dotyczącego okresu eksploatacji dyskietek. Z maksymalnych wartości tego okresu, jakie często podają producenci, nie należy jednak wyciągać uproszczonych wniosków. Dyskietka o krótkim okresie żywotności nie zawsze jest jakościowo gorsza.

Podstawowymi kryteriami oceny własności mechanicznych dyskietki są jakość jej powierzchni oraz zachowanie się pod wpływem zmian temperatury. Jest pewne, że szorstka powierzchnia prowadzi do szybszego zużycia dyskietki oraz zanieczyszczenia głowic. Należy przypomnieć, że na dyskietce dane są zapisywane w postaci koncentrycznie rozmieszczonych i podzielonych na sektory ścieżek magnetycznych. W badanych dyskietkach o średnicy 130 mm powszechnie występują gęstości 1,9 ścieżki/mm (48 ścieżek na cal) oraz 3,8 ścieżki/mm (96 ścieżek na cal). Jak już wspomniano, obszar wyznaczony do zapisu ma szerokość mniejszą niż jeden cal, co oznacza, że na dyskietce znajduje się 40 lub 80 ścieżek, numerowanych zazwyczaj w kierunku do środka dyskietki: ścieżka 0 znajduje się więc na okręgu o promieniu największym, natomiast 39 lub 79 – o promieniu najmniejszym, co oznacza, że również długość kolejnych ścieżek zmniejsza się w kierunku do wewnątrz. Pojedyncza ścieżka ma szerokość 0,33 mm (40 ścieżek) lub 0,16 mm (80 ścieżek). Przy maksymalnej gęstości liniowej 5500 bitów/cal (podwójna gęstość zapisu), na 1 mm ścieżki mieści się 216 bitów. Tak więc dla dyskietki np. komputera IBM PC jeden bit na najkrótszej ścieżce zajmuje obszar długości zaledwie 5 µm i szerokości 330 µm. W komputerze Sirius 1 warunki zapisu są jeszcze bardziej krytyczne, ponieważ jeden bit musi się tam zadowolić obszarem 3×160 µm. Dlatego też niejednolitość powierzchni magnetycznej dyskietki w stosunku do szerokości ścieżki powinna być jak najmniejsza.

WPŁYW TEMPERATURY

Dyskietki powinny być eksploatowane w zakresie temperatur od 10 do 50 C. Jeśli jednak 80-ścieżkową minidyskietkę zapiszemy w temperaturze 10°C, ■ odczytamy przy 50°C, to ścieżka rozszerzy się o ok 25%. Zakładając próg możliwości odczytu wynoszący 50% sygnału wzorcowego, to w sytuacji gdy głowica wskutek rozregulowa-

nia zmieni swoje położenie o 25%, dyskietka nie zostanie odczytana. Doświadczenia wskazują, że w pracy z dyskietkami należy unikać temperatur granicznych. Firma Nashua zaleca np., aby dyskietkę przed użyciem przetrzymać w temperaturze pokojowej co najmniej jedną godzinę. Dla dyskietek przeznaczonych do regulacji napędu firma Dysan zaleca nawet 24-godzinny okres aklimatyzacji.

BADANIA

Badaniem objęto minidyskietki następujących dwunastu producentów (w nawiasach kraj pochodzenia): BASF (RFN), Döbbelin & Böder (RFN), 3M Scotch (Włochy), Dysan (USA), Elephant Memory Systems (USA), Fuji (Japonia), Maxell (Japonia), Memorex (Irlandia), Rhone Poulenc Systems (Francja), TDK (Japonia), Verbatim (USA) oraz Xidex (USA). Jako próbki pobrano po jednej dyskietce z oryginalnych opakowań zawierających po 10 sztuk. W wypadku stwierdzenia widocznych różnic w wyglądzie, próbkę porównano z innymi egzemplarzami tego samego opakowania, a nawet innych opakowań. Wybrano dyskietki 80-ścieżkowe z zapisem jednostronnym lub dwustronnym, badając jednak wyłącznie stronę 0. Wyniki badań można bez zastrzeżeń przenieść na częściej sprzedawane dyskietki 40-ścieżkowe.

Własnosci mechaniczne zbadanych dyskietek ujęto w tabeli. Podana w mm grubość dotyczy obwoluty ochronnej. Grubość całkowita wynosząca od 1,3 do 1,6 mm była zgodna z obowiązującymi normami. Chociaż nie stwierdzono większych różnic, dyskietki niektórych firm wydawały się w subiektywnym odczuciu wyraźnie odporniejsze na zginanie. Do mniej sztywnych zaliczono dyskietki firm Fuji, Maxell, TDK oraz BASF. W tym ostatnim wypadku przyczyną mniejszej sztywności jest prawdopodobnie dwuczęściowa otulina. Do bardziej sztywnych zaliczono wyroby firm 3M, Verbatim oraz Xidex.

W tabeli określono rodzaj materiału obwoluty ochronnej oraz jakość spawania. Stwierdzono, że wszystkie obwoluty były poprawnie zespawane, oraz że trwalsze okazało się spawanie całej powierzchni zagięcia.

Istotnym elementem usztywniającym dyskietkę jest pierścień przyklejony wokół jej otworu środkowego na stronie 1. Poprawia on stabilność mocowania dyskietki w napędzie i zapobiega przedwczesnemu wyszczerbieniu otworu srodkowego, czyli utracie osiowości. Zabezpieczenie to jest szczególnie potrzebne przy eksploatacji dyskietek w niestarannie wykonanych, tanich napędach komputerów domowych.

Miarą jakości wypolerowania obszaru zapisu jest jego połysk. Dyskietki oceniano pod tym względem w skali pięciostopniowej. najbardziej lśniącym przyznawano wartość 5.

Szumy powstające podczas wirowania wskutek ruchów otuliny i foiii magnetycznej są w poszczególnych wyrobach dość zróżnicowane. Nie stwierdzono jednak, aby miały one wpływ na jakość zapamiętywania danych. Do najgłośniejszych należały wyroby firm Dysan oraz Elephant Memory.

Badania mikroskopowe powierzchni dyskietek przeprowadzono w Instytucie Fizyki Stosowanej Wyższej Szkoły Technicznej w Darmstadt za pomocą mikroskopu elektronowego rastrowego typu JEOL JSM U3. Jako próbkę wycinano z każdej dyskietki fragment obszaru zapisu, który analizowano za pomocą powiększeń 2400 oraz 800 razy. W wypadku stwierdzenia w obrazie mikroskopowym znacznej niejednolitości struktury powłoki magnetycznej, stosowano dodatkowo mniejsze powiększenie, pozwalające ocenić rozłożenie nierówności powłoki na większym obszarze powierzchni dyskietki.

Porównanie zdjęć mikroskopowych doprowadziło do następujących stwierdzeń.

■ W wyrobach firm BASF, D & B, 3M oraz Xidex powłoka magnetyczna ma charakter wzorcowy. W dyskietkach firmy 3M można to zaobserwować nawet gołym okiem w postaci koncentrycznych pierścieni, a badania dwóch różnych partii wykazały identyczny rezultat. Struktury wzorcowe powstają wskutek intensywnego polerowania, prowadzącego do uzyskania wyjątkowo gładkiej powierzchni oraz połysku. Zakładając, że dla lepszego kontaktu głowicy z dyskietką istotny wpływ ma gładsza powierzchnia, wyroby o takich cechach powinny lepiej zachowywać się podczas eksploatacji. Jednakże nadmiernie gładkie powierzchnie mogą powodować występowanie tzw. efektu „slid & slide" (skoczyć i poślizgnąć się). Efekt taki występuje jednak tylko niektórych napędach i charakteryzuje się cyklicznym, któtkotrwałym oddalaniem się głowicy od powierzchni dyskietki, co oczywiście powoduje błędy zapisu. Ten rodzaj usterek nie wystąpił podczas testu w żadnej z badanych dyskietek.

■ Niektóre dyskietki wykazują znaczną niejednolitość struktury powłoki magnetycznej. Nierówności uwidoczniły się wyraznie na zdjęciach mikroskopowych jako ciemniejsze plamy w dyskietkach firm BASF, D & B, 3M, Maxell, Memorex, TDK oraz Xidex. Celem sprawdzenia, czy nierówności te mają charakter miejscowy, czy tez występują na całej powierzchni dyskietki, próbki sfotografowano powtórnie w mniejszym powiększeniu. Ocena nierówności z punktu widzenia wartości użytkowej dyskietki nie jest jednak prosta. Przy szerokości ścieżek 0,33 mm (40-ścieżkowa) żadna ze wspomnianych dyskietek nie powinna stwarzać trudności eksploatacyjnych. Trudności takie mogą jednak wystąpić przy szerokości ścieżki 0,16 mm (80-ścieżkowa) wtedy, gdy pojedyncze zagłębienia przekraczają 1/4 szerokości ścieżki. Zagłębień tak dużych w badanych próbkach jednak nie stwierdzono, a większosc z nich mieściła się w granicach 10-25 µm. Potwierdził to test, podczas którego wszystkie badane dyskietki zapisano bezbłędnie na wszystkich osiemdziesięciu ścieżkach.

■ Mikroskopowy obraz dyskietek o najmniejszym połysku nie zawsze wskazywał na nierównomierność struktury powłoki magnetycznej.

micro

Podsumowując wyniki badania własności mechanicznych można stwierdzić, że z punktu widzenia struktury powłoki magnetycznej najlepszą ocenę należy przyznać dyskietkom firm Dysan, Elephants Memory, Fuji oraz Rhone Poulenc. Interpretując wyniki badań należy jednak pamiętać, że są to tylko fragmenty bardzo skomplikowanych zależności. O cechach użytkowych dyskietki decyduje bowiem wiele czynników, często bardziej zależnych od napędu, niż od samej dyskietki.

Przybliżoną całościową ocenę badanych dyskietek można będzie jednak sformułować dopiero po uwzględnieniu ich własności magnetycznych, które zostaną omówione w drugiej części artykułu.

mikroklan

Mniej znane możliwości asemblera **GENS3**

Każdy kto wykorzystuje asembler GENS3 dla mikroprocesora Z80 korzysta z pewnego zestawu dyrektyw ogólnie znanych. Nie wszyscy jednak znają pełny wachlarz jego możliwości.

OPCJE WYPISYWANE W TREŚCI PROGRAMU

W polu etykiety należy umieścić następującą sekwencję:

★ litera opcji>< [<+|–>] [<ciąg znaków>]

Litery opcji:

L – służy do włączania i wyłączania wydruku (+ włącza, –wyłącza) w strumieniu 2 (na ogół jest to ekran, czyli kanał S). Wartość domyślna tej opcji (tzn. jej stan), jeśli jej nie podano w ogóle, zależy od ustawienia drugiego bitu w bajcie opcji asemblacji.

D – wybiera bazę, według której będą drukowane adresy na wydruku asemblacji (+ dziesiętnie, – szesnastkowo).

C – włącza drukowanie kodu wynikowego na wydruku asemblacji (+ kod jest drukowany, – kod nie jest drukowany).

S – jeśli jest włączony wydruk opcją ★L+, to asembler wstrzymuje działanie i czeka na naciśnięcie dowolnego klawisza. Opcja ta może być przydatna do podglądania określonych punktów kluczowych na wydruku. Parametry nie są wymagane.

E – powoduje wypisanie na wydruku asemblacji ciągu znaków występujących za literą opcji oraz wygenerowanie kilku wierszy odstępu.

F – powoduje odczytanie z taśmy przez bufor i asemblację tekstu źródłowego.

Po wprowadzeniu przez użytkownika dyrektywy *A* (asembluj), program pyta o rozmiary tablicy symboli oraz bajt opcji. Manipulując odpowiednio tymi wielkościami można wykorzystać dość duże możliwości asemblacji.

Znaczenie poszczególnych bitów w bajcie opcji zawiera tabela.

Bit	Znaczenie
0	Wydruk tablicy symboli po zakonczeniu asemblacji
1	Niezaplsywanie kodu wynikowego w pamlęci komputera
2	Niedrukowanie wydruku asemblacji
3	Wysłanie wydruku asemblacji na strumień 3 (zazwyczaj drukarka)
4	Umieszczenie kodu wynikowego za tablicą symboli

KOD WYNIKOWY

Główna różnica w korzystaniu z asemblera GENS3 w porównaniu do typowych asemblerów, pracujących np. pod nadzorem systemu operacyjnego CP/M, polega na niekorzystaniu z programów LINKER i LOADER. Nie ma tu zatem możliwości niezależnej kompilacji oraz ewentualnego łączenia bloków kodu przez tablice symboli.

Cały program wynikowy musi być w zasadzie generowany jednokrotnym wykonaniem asemblacji. Taka sytuacja oznacza, że przy większych programach (zawierających rozbudowane tablice symboli) program wynikowy może nie mieścić się w docelowym miejscu pamięci.

Istnieją dwa wzajemnie uzupełniające się sposoby rozwiązywania problemu braku dostatecznej pojemności pamięci operacyjnej. Należy jednak podkreślić, że problemu tego nie można całkowicie pominąć, gdyż ze względu na określoną pojemność pamięci komputera, nawet gdyby można asemblować cały program wprowadzany z taśmy, a następnie umieszczać na niej cały kod wynikowy, to i tak w pewnym momencie zabrakłoby miejsca na tablicę symboli. Sposoby te obejmują:

- umieszczanie kodu wynikowego otrzymywanego w trakcie asemblacji nie w miejscu docelowym, lecz za tablicą symboli,
- korzystanie z możliwości asemblacji tekstu źródłowego wprowadzanego wprost z taśmy przez dyrektywę ★*F*.

Załóżmy, że początek tekstu źródłowego znajduje się pod adresem 32256, asembler – pod adresem 24500, tekst źródłowy ma długość 20 KB, tablica symboli ok. 7000 bajtów, a przygotowany program ma być wykonywany od adresu 32768 (32 KB szybszej części pamięci RAM). Program ten należy pisać tak, jakby miał się on znaleźć w pamięci pod adresem 32768 (tzn. rozpocząć dyrektywą ORG 32768). Za pomocą dyrektywy *X* należy odczytać adres końca tekstu (jest to druga z podanych liczb).

Po wprowadzeniu dyrektywy *A* (asembluj) należy podać wielkość tablicy symboli, którą oblicza się podając adres, gdzie ma być umieszczony kod wynikowy, od którego należy odjąć liczbę *2* oraz liczbę będącą adresem końca tekstu.

W podanym przykładzie rozmiar tablicy symboli będzie równy 60000-2-52736. Następnie wybiera się taką opcję asemblacji, aby czwarty bit został włączony (na przykład opcję 20).

Po wykonaniu asemblacji należy zapisać na taśmie kod wynikowy spod adresu 60000. Jest on przygotowany do pracy pod adresem 32768. Można go wczytać do pamięci dyrektywę *LOAD "kod" CODE 32768* oraz uruchomić instrukcję *RANDOMIZE USR 32768*. Jeśli program nie otwiera sam strumienia *2*, to należy go inicjować instrukcją *PRINT USR 32768*. Wówczas system ma ustawiony strumień jako aktualny, a wydruki z programu pojawią się na ekranie. Sposobem tym można również generować kod maszynowy przystosowany do pracy na innym komputerze, z pamięcią RAM dostępną od adresu 0. Jeśli tekst źródłowy jest tak długi, że opisany sposób nie spełnia zadania, to można skorzystać z opcji ★*F*.

W celu asemblacji tekstu źródłowego wprowadzonego wprost z taśmy należy:

- Podzielić tekst źródłowy na części i zdecydować się, które są już mniej więcej ustalone (chodzi o to, aby nie zmieniać zbyt często ich zawartości, gdyż jest to dość pracochłonne).
- Nagrać dyrektywą *P* każdą z ustalonych części oddzielnie. Po nagraniu na taśmie z tekstami źródłowymi, należy je jeszcze nagrać na oddzielnej taśmie. Nagranie to wykonywane jest dyrektywą *T*. Dzieli ona program na bloki o takich rozmiarach jak wielkość bufora, o którą pyta początkowo asembler. Moduły te nagrywa po kolei, oddzielając je przerwą z tonem synchronizacyjnym. Dyrektywa *T* wymaga jako parametrów numeru wiersza, od którego chcemy zacząć, oraz numeru wiersza, na którym zamierzamy skończyć nagranie. Jedną z tych części należy potraktować jako „główną" w tym sensie, że będzie ona sterować odczytywaniem fragmentów tekstu z taśmy. W odpowiednich miejscach części głównej należy umieścić opcję asemblacji: *F tytuł fragmentu*. Główną część należy nagrać na taśmie tylko dyrektywą *P*.

W czasie asemblacji w pamięci operacyjnej musi się zmieścić tylko moduł sterujący. tablica symboli i kod wynikowy.

Przy pisaniu programów, które będą asemblowane z opcją bitu *4*, tzn. z umieszczaniem kodu za tablicą symboli, należy mieć na uwadze fakt, że dyrektywa asemblera *ORG* nie powoduje pominięcia odpowiedniej liczby komórek pamięci. Należy w tym celu stosować dyrektywę *DEFS*.

Przykład:

```
ORG #0
DI
JP RSTCN
ORG #8
; obsługa RST 8
```

pokazuje zastosowanie niewłaściwe dyrektywy ORG.

Powinno być:

```
ORG #0
DI
JP RSTCN
DEFS #8–$ ; $ jest aktualną wartością licznika rozkazów
```

Pierwszy wariant programu nie wstawiłby niezbędnych zer między *JP* a adresem *8*.

PIOTR PARLEWICZ

SPOKREWNIONE Z WYBORU

CZYLI

KOMPATYBILNE Z WŁASNEJ WOLI

Oferta producentów mikrokomputerów kompatybilnych z IBM PC/XT jest obecnie olbrzymia i stale trudna do przeanalizowania. Miesięcznik „Micro" zamieścił zestawienie obejmujące aż 40 modeli, w których stwierdzono pełną kompatybilność. Aby ułatwić orientację naszym Czytelnikom, z zestawienia tego wybraliśmy tylko bardziej znanych w Polsce „krewnych" IBM PC/XT, zwracając uwagę na ich główne parametry.

W 1981 roku zmienił się świat mikrokomputerów. IBM wkroczył na rynek i stworzył swoim PC pewien standard. Bardzo wielu dużych i małych producentów podczepiło się do IBM-owskiego pociągu, oferując prawie identycznie skonstruowany, tak zwany sprzęt kompatybilny. Ostatnio, w celu zakamuflowania podobieństwa do IBM PC/XT, zaczęto zastępować kompatybilność terminem standard przemysłowy. Profesjonalny rynek komputerowy jest w znacznym stopniu opanowany przez mikrokomputery tej klasy.

Nie jest natomiast łatwo zdobyć znaczącą pozycję rynkową takim producentom, jak np. Apple, którzy usiłują przeforsować własną koncepcję. Mimo wielu innowacji firmie tej nie udało się do tej pory wejść na rynek zastosowań profesjonalnych. Commodore – będący obok Appie na rynku mikrokomputerowym głównym antagonistą IBM-owskiego giganta – przyłączył się również do linii IBM.

Z chwilą, gdy na jesieni 1984 roku IBM zaprezentował nowy model PC/AT (ang. Advanced Technology – zaawansowana technologia) – zarysowała się możliwość przełomu. Zaczęto mówić o klęsce kompatybilnych i o ciężkich dla nich czasach. Były to jednak mylne przepowiednie. Jednostanowiskowe PC mają w dalszym ciągu rację bytu, zwłaszcza że IBM miał od początku problemy techniczne z PC/AT i do dziasiaj nie może w zadowalający sposób rozwiązać wynikłych z tego powodu trudności w dostawach. Zresztą, ku pełnemu zadowoleniu konkurencji.

W tym samym czasie firmy Compaq, Texas Instruments i Kaypro przedstawiły pierwsze kompatybilne PC/AT, likwidując tym samym problem opóźnienia w oferowaniu tych systemów.

Na co powinien zwrócić uwagę użytkownik, decydując się na zakup mikrokomputera kompatybilnego z PC? Są przecież producenci, których wyroby nie są wcale kompatybilne lub tylko częściowo, a mimo to próbują płynąć na fali kompatybilności. Poza tym, „kompatybilny" nie jest wcale równoważny z „dobrym". Tylko częściowo kompatybilne, a mimo to świetne, są mikrokomputery osobiste takich firm, jak Altos, Hewlett Packard, Sanyo, Sharp czy Siemens.

Na określenie kompatybilności operacyjnej zasługują wszystkie PC, które spełniają następujące wymogi:

- może przenieść z IBM PC prawie wszystkie programy aplikacyjne,
- można zainstalować wszystkie rozszerzenia sprzętowe IBM PC,

■ ekran monitora, klawiatura i dokumentacja odpowiadają rozwiązaniom PC.

Jako funkcjonalnie kompatybilne określa się takie mikrokomputery, które mogą czytać zbiory z dyskietek IBM-owskich, zapisywać zbiory na tych dyskietkach oraz wykonywac najważniejsze (specjalnie przystosowane) programy standardowe MS-DOS.

Najniższy poziom kompatybilności osiąga się wtedy, gdy mikrokomputer kompatybilny z PC może czytać dyskietki z danymi zapisanymi na IBM PC.

Zdaniem amerykańskiego czasopisma „PC-World" mikrokomputery kompatybilne z IBM PC powinna cechować zgodność poniższych kryteriów.

Procesor. IBM PC jest oparty na mikroprocesorze Intel 8088. Mikrokomputery kompatybilne powinny wykorzystywać taki sam procesor. Pojawiły się już mikrokomputery, w których zastosowano następców 8088 (8086, 80188 lub 80186). Procesory te wykonują wszystkie rozkazy 8088, są jednak znacznie szybsze. Mogą więc wystąpić pro-

Producent Model	IBM IBM PC	IBM IBM PC/AT	Commodore PC-10 (PC-20)	Compaq Computer Deskpro 286 (portable 286)	Compaq Computer Deskpro (portable)
Procesor **Częstotliwość (MHz)**	8088 4,77	80286 6	8088 4,77	80286 ■ lub 8	8086 4,77 lub 8
Pamięć operacyjna: **– standardowa (KB)** **– maksymalna**	64 640	256 3000	256 640	256/512 (640) 8.200 (2,600)	256 640
Interfejsy: **– równoległe** **– szeregowe** **– IEC-Bus**	1 1 opcja	1 1 opcja	1 1 –	1 1 –	1 1 –
Dyskietki: **– średnica (cale)** **– pojemność (kB)** **Dysk sztywny (MB)**	5 1/4 360 20	5 1/4 1200 2×20	5 1/4 360 10	5 1/4 1×200 30,70 (20 opcja)	5 1/4 360 10,30
Ekran: **– wielkość (cale)** **– liczba wierszy** **– liczba znaków w wierszu** **– rozdzielczość** **– możliwość grafiki**	11,5 25 80 720 × 350 tak	11,5 25 80 720 ×350	12 25 80 tak	12 (9) 25 80 maks. 720 × 350 barwna jako standard	12 (9) 25 80 maks. 720 × 350 barwna jako standard
Klawiatura: **liczba klawiszy** **– funkcyjnych** **– programowanych**	83 10 10	83 10 10	85 20 w dwóch płaszczyznach 20 ■ dwóch płaszczyznach	84 10 30	83 10 30
System operacyjny	PC-DOS 3.0	PC-DOS 3.0	MS-DOS 2.11	MS-DOS 3	MS-DOS 2
Różnice ■ stosunku ■ IBM PC/XT				30% szybszy opcje: – zintegrowana jednostka taśmowa – zintegrowany dysk stały (20, 30, 70 MB)	przy pełnej kompatybilności 2-3 razy szybszy

blemy, ale tylko w bardzo nielicznych zastosowaniach, uwzględniających przebiegi czasowe, to znaczy wtedy, gdy program oczekuje wolniejszego procesora, np. Intel 8088.

Struktura systemu. IBM PC ma specyficzną strukturę systemu, co przykładowo znajduje swój wyraz w układzie pamięci RAM i ROM oraz przestrzeni adresowej wejścia-wyjścia. Struktura ta powinna być odwzorowana w identycznej postaci w kompatybilnych mikrokomputerach, po to, aby zagwarantować możliwość eksploatacji oprogramowania zależnego bezpośrednio od cech sprzętu.

Odrębny problem stanowi tak zwany ROM-BIOS. Terminem tym określa się pewną liczbę wysoce efektywnych procedur systemowych, zapisanych w pamięci ROM mikrokomputera IBM PC, do których odwołują się bezpośrednio programy standardowe. Ponieważ procedury te są chronione prawami autorskimi IBM (copyright), producenci kompatybilnych mikrokomputerów nie mogą ich po prostu skopiować. Jednakże wielu naśladowcom IBM udało się już odwzorować ROM-BIOS IBM PC bez naruszenia praw autorskich IBM.

Data General DG/One	Honeywell Bull Bull Micral 30	Kaypro Europe Kaypro 286	Olivetti M21 (M24)	Sperry Sperry PC
8088 4	8088 4,77	80286 6	8088 8	8088 4,77 i 7,18
256 512	128 640	512 840 (1.500)	128 640	258 640
2 – –	1 1 –	2 2 –	1 1 –	1 1 opcja
3 1/2 (opcja 5 1/4) 720 10 (opcja)	5 1/4 2 × 360 10 (opcja)	5 1/4 2 × 200 10 (20 opcja)	5 1/4 1 lub 2 × 320, 360, 640, 720 10	5 1/4 360 10 (20 opcja)
19 × 28 cm, LCD 25 80 640 × 256 tak	12 25 80 720 × 350 tak (320 × 200)	wg życzeń użytkownika 840 × 200 barwna jako standard	9 (12) 26 80 640 × 400 tak	12 25 80 840 × 400 barwna jako standard
81 10 	83 10 10	84 10 10	83 opcja 10 (18 opcja) 10	84 10 10 × 2
MS-DOS 2.11	MS-DOS 2.11	DOS 2.11	MS-DOS 2.11	
przenośny, duży ekran	dodatkowy system operacyjny „Prologue"; zintegrowane jednostki pamięci		8086/80286; brak pamięci dynamicznej	lepsze możliwości graficzne; Caps/Num Lock podświetlany; 5 dodatkowych gniazd

Gniazda krawędziowe (ang. slots). W IBM PC znajduje się wiele wolnych miejsc, przeznaczonych do umieszczenia dodatkowych pakietów, rozszerzających możliwości sprzętowe danej konfiguracji. Kompatybilne mikrokomputery powinny również umożliwiać wstawienie dodatkowych pakietów, przeznaczonych dla IBM PC.

Dyskietki. Stosowane w IBM PC dyskietki o średnicy 5 1/4" są zapisywane przez poszczególnych producentów w najróżniejszych formatach, co powoduje, że są często między sobą niekompatybilne. W IBM PC występują cztery różne formaty zapisu (osiem lub dziewięć sektorów na ścieżce oraz zapis jedno- lub dwustronny), ktore powinny być odczytane przez mikrokomputer kompatybilny tak samo, jak są odczytywane na IBM PC.

Monitor ekranowy. W celu przyspieszenia tworzenia obrazu na ekranie monitora, w IBM PC istnieje możliwość bezpośredniego dostępu do pamięci ekranu. Również mikrokomputery kompatybilne powinny zapewniać taką możliwość, przy czym struktura i adresy pamięci muszą być takie same jak w IBM PC. Identyczne powinny być również rozkazy ekranowe, np. adresowanie kursora.

Zestaw znaków i klawiatura. W IBM PC występuje wiele specyficznych rozwiązań, np. znaczna liczba znaków specjalnych odwzorowywanych na ekranie oraz nietypowa budowa klawiatury. Dlatego też mikrokomputery kompatybilne powinny być identycznie zbudowane, co jednak nie wyklucza drobnych usprawnień, przykładowo w rozplanowaniu klawiatury.

System operacyjny. System operacyjny IBM PC-DOS jest identyczny z systemem MS-DOS firmy Microsoft. Mikrokomputer kompatybilny powinien więc byc wyposażony w ten system operacyjny, w miarę możliwości jego najnowszą wersję (2.1 lub następną). Ponadto, firma Digital Research opracowała wersję systemu CP/M 86, dostosowaną do IBM PC. Mikrokomputer kompatybilny powinien również pracować pod nadzorem tego systemu.

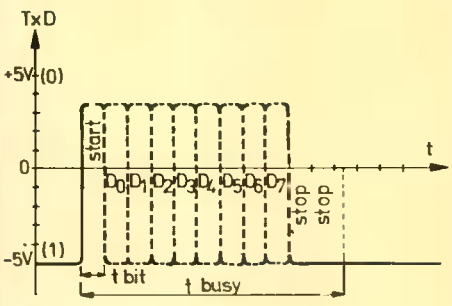
micro

KONWERTER CENTRONICS RS–232C

Konwerter Centronics RS-232C został skonstruowany z sześciu układów scalonych TTL na płytce o wymiarach 90×80 mm (bez zasilacza). Umożliwia to łączenie drukarki złączem szeregowym, np. drukarki Seikosha GP-500AS, z mikrokomputerem Amstrad CPC-664 lub CPC-6128. Konwerter daje możliwość programowego ustawiania ósmego bitu danych (w Amstradzie jest on pomijany), umożliwiając transmisję prawie wszystkich 254 kodów ASCII (z wyjątkiem kodów 127 i 255). Opcja ta pozwala wykorzystywać drukarkę w trybie graficznym. Szybkość transmisji wynosi 1200 bodów.

UKŁAD ELEKTRONICZNY

Siedmiobitowa szyna danych złącza Centronics *D0-D6* wchodzi na wejścia równoległe *A-G* rejestru przesuwnego *IC1 SN74199*, (można także, po zmianie połączeń, użyć układów *198*, *299*, *298*, *165* lub *166*). Wejście *H* (uziemione) definiuje bit startowy w standardzie RS-232. Na wejście szeregowe *J-K* tego układu podany jest sygnał z wyjścia *Q2* przerzutnika *2* w *IC4* definiujący ósmy bit danych. Bit ten jest kasowany po włączeniu zasilania dzięki użyciu kondensatora *C4*. Może on zmienić swój stan, o ile podczas przyjścia impulsu strobującego *STRB/* na wejściach *D0-D6* wystąpią same jednostki (kod 127). Sytuacja taka prowadzi przy tym do blokady przerzutnika *1*



Rys. 1. Sygnał RS-232C

w mikroukładzie IC4, co sprawia, że kod 127 pomimo wygenerowania sygnału BUSY, nie jest transmitowany przez wyjście TxD konwertera. Kod sterujący może, w razie potrzeby, zostać przedefiniowany przez zastąpienie bramki *IC2* bardziej rozbudowanym układem logicznym (stosunkowo prosta wymiana na układ *CMOS 4078* wraz z zamianą połączeń *J-K* między przerzutnikami *1* i *2* w *IC4* spowodowałaby zmianę kodu na kod *0*).

Czas trwania impulsu dla pojedynczego bitu na wyjściu RS-232 jest określony stałą *R1C1* i wynosi w tym wypadku: *tbit = 0.3★R1★C1 = 0.83 ms*. Czas trwania impulsu BUSY na wyjściu Centronics jest natomiast zdefiniowany przez stałą *R2C2* i wynosi *tbusy = 0.3★R2★C2 = 9.58 ms*. Ważne jest, aby był spełniony warunek: *tbusy > 11★tbit*, dlatego też przy uruchamianiu niezbędny jest oscyloskop. Zamiast oporników *R1* i *R2* można zastosować dobrej klasy potencjometry montażowe rezystancji 250 kΩ.

Przerzutnik monostabilny *1* w *IC5 SN74LS123* jest połączony w układzie samowzbudnym (czas przerwy jest określony przez kondensator *C3* i wynosi ok. 200 ns – uwaga: usunięcie uniemożliwiłoby prawidłowe wysterowanie wejścia *CLK2* układu *74199*). Przerzutnik ten, sprzężony z dzielnikiem *IC3 SN74LS93* powoduje wygenerowanie dokładnie dziesięciu impulsów za każdym pojawieniem się sygnału strobującego STRB/ (niski poziom) na wejściu Centronics. Te dziesięć impulsów powoduje przesuwanie zawartości rejestru *IC1* i pojawienie się kolejno na wyjściu RS-232 bitu startowego, ośmiu bitów danych oraz co najmniej dwóch bitów stopu (patrz rysunek 1).

Napięcie –5V potrzebne do wysterowania wyjścia *TxD* RS-232 jest wytwarzane przez prostą przetwornicę opartą na tranzystorze *T3*. Uzwojenie transformatora *TF1* (o indukcyjności 0.5 mH) jest nawinięte bifilarnie drutem *DNE* średnicy *0,15 mm* na rdzeniu toroidalnym *F81* (lepiej użyc innego rdzenia o większej przenikalności) ◘ średnicy zewnętrznej *10 mm* i zawiera 100 zwojów. Częstotliwość pracy wynosi ok. 100 kHz. Wykonanie transformatora nie jest krytyczne i układ będzie pracował praktycznie z każdym.

Możliwa jest również prosta modyfikacja układu przez wyprowadzenie sygnałów *D0-D6*, *STRB/*, *BUSY* oraz *D7* (patrz linia przerywana na rys. 2) na dodatkowe gniazdo zewnętrzne, umożliwiające wysterowanie drukarek z 8-bitowym wejściem Centronics.

OPIS KONSTRUKCJI

Układ został wykonany na płytce drukowanej jednostronnej o wymiarach 80×90 mm, z połączeniami częściowo poprowadzonymi przewodami. Z jednej strony płytki wlutowane jest gniazdo do złącza krawędziowego w mikrokomputerze, natomiast wyjście RS-232 jest wyprowadzone kablem dwużyłowym ekranowanym do 25-stykowej wtyczki Canon typu „male". Do zasilania wykorzystano seryjny zasilacz kalkulatorowy *ZS 02/6/1* (z diodą Zenera wymienioną na *5,6 V*), połączony kablem z płytką konwertera. Obudowę o wysokości 20 mm zrobiono ze zlutowanych, a następnie pomalowanych, kawałków laminatu miedziowanego, wlutowując do niej płytkę konwertera.

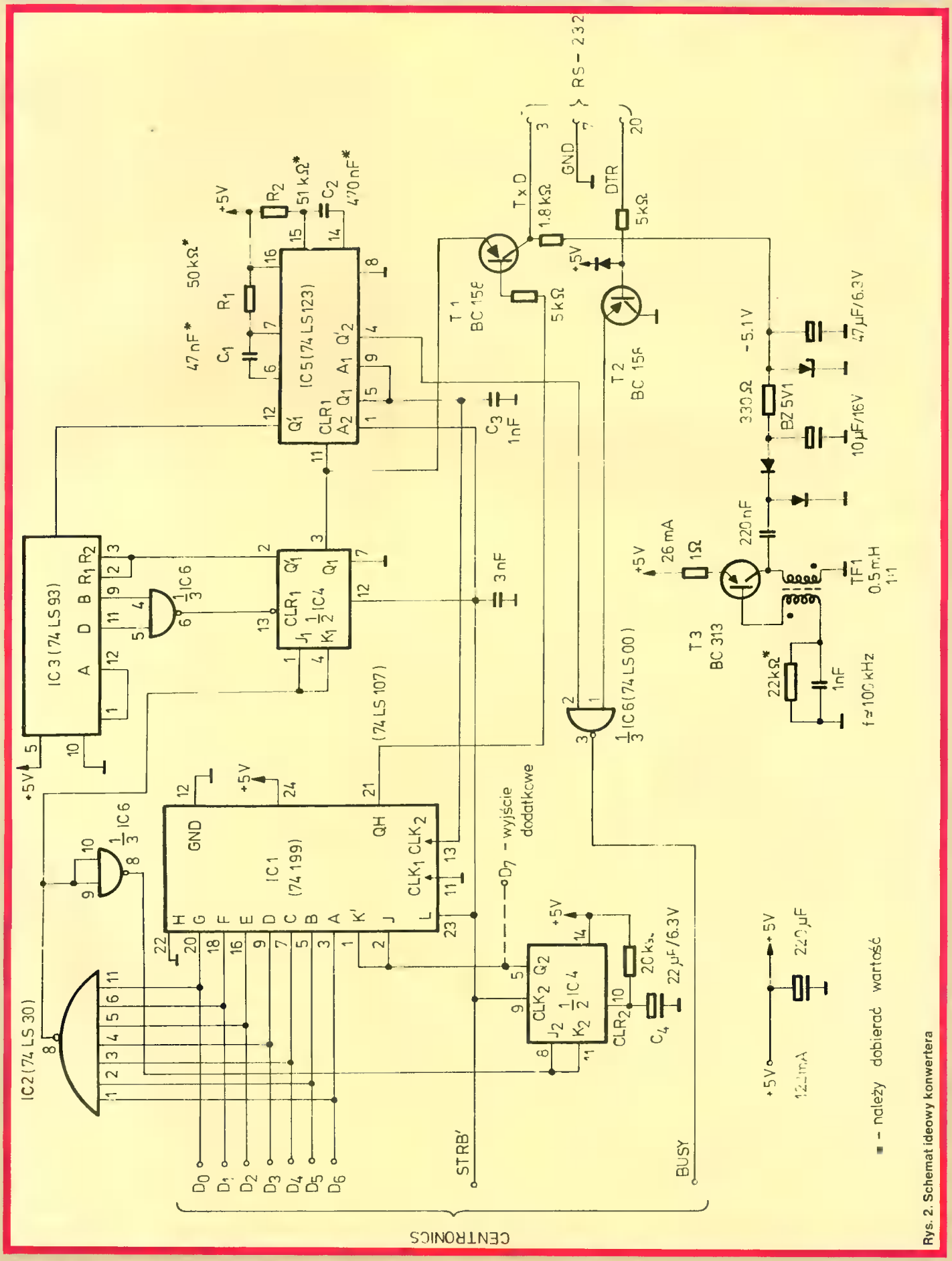
Podstawowe dane techniczne konwertera:

Napięcie zasilania: 5 V ±10%
Pobór prądu zasilania: 122 mA
Wejścia:
Centronics: 7 bitów danych (poziom TTL) oraz sygnał strobujący STRB/ (TTL) o czasie trwania od 200 ns do 0.83 ms,
RS-232: DTR (Data Terminal Ready)
Wyjścia:
Centronics: BUSY (TTL-LS)
RS-232: TxD (Receiver Data) ok. +3.5 V (logiczne 0), –5 V (logiczne 1), prąd zwarcia – kilka mA
Szybkość transmisji danych: 1200 bitów/s

Kod sterujący konwerterem *127* (nie transmitowany na wyjście TxD pomimo aktywnego sygnału BUSY) powoduje przełączenie ósmego bitu danych na wyjściu RS-232 na wartość logiczną przeciwną do poprzedniej. W momencie włączenia zasilania bit ten jest kasowany.

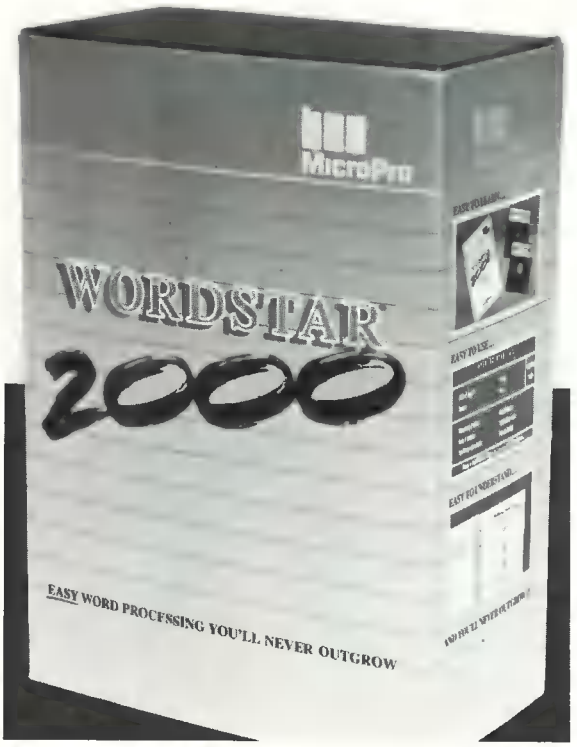
Warto wspomnieć, że podobny efekt można uzyskać przenosząc całą komplikację do sfery oprogramowania. Wtedy wystarczy zbuforować (na wszelki wypadek) jeden z bitów danych i sygnał BUSY. Sygnały te można podłączyć do drukarki jako TxD i DTR. Takie rozwiązanie wymaga napisania własnego modułu programu obsługi drukarki szeregowej (tzw. device driver).

STANISŁAW BŁESZYŃSKI



Rys. 2. Schemat ideowy konwertera

WordStar 2000
czyli
nowe szaty króla



Jeszcze kilka lat temu WordStar, opracowany przez firmę MicroPro, był jednym z najlepszych programów przetwarzania tekstów. Konkurencja jednak nie spała i w krótkim czasie pojawiły się programy znacznie prostsze w użyciu. Aby sprostać zwiększonym wymaganiom użytkowników opracowano nową wersję programu pod nazwą WordStar 2000. WordStar zaliczany był do procesorów tekstów o średniej wydajności, natomiast WordStar 2000 jest już niewątpliwie programem najwyższej klasy, takim jak MS Word i Word Perfect 4.0.

Pierwszy kontakt z programem wygląda bardzo zachęcająco. Poznanie programu opiera się na konwersacji z komputerem za pomocą ekranu i klawiatury. Dla nowicjuszy jest to idealne wprowadzenie w temat, dla doświadczonych – przyjemna zabawa. Rozwiązanie takie eliminuje żmudne wertowanie podręcznika i wyszukiwanie potrzebnych rozkazów, których pełny zestaw jest dostarczony na składanym kartoniku. Specjalna nakładka na klawiaturę opisuje natomiast zadania klawiszy funkcyjnych. Jeżeli mimo to pojawią się pytania, przez 90 dni można korzystać z konsultacji telefonicznych. Osoby, które zechcą szybciej i głębiej poznać program, mogą korzystać z indywidualnego szkolenia.

WordStar 2000 dostarczany jest na pięciu dyskietkach i z wyglądu przypomina nieco zintegrowany pakiet programowy Lotus 1-2-3.

INSTALOWANIE

Na pierwszych stronach podręcznika opisano podstawowe sprawy: włączanie komputera, formowanie dyskietki, wykonanie kopii. Podano też, jak zainstalować program, czyli jak dopasować go do posiadanego sprzętu.

Instalowanie odbywa się w wygodnym dla użytkownika trybie konwersacyjnym. Do zapamiętania wprowadzonych danych potrzebna jest pomocnicza dyskietka, ponieważ po zainstalowaniu programu na dyskietce nie ma już wolnego miejsca.

Jeszcze niedawno jedynie zintegrowane pakiety programowe wymagały zwiększenia pojemności pamięci operacyjnej, co prowadziło do konieczności dołączenia dysku sztywnego. Obecnie już jeden program przetwarzania tekstu wykorzystuje do końca przeciętną konfigurację mikrokomputera. Nie oznacza to jednak, że WordStar 2000 można stosować tylko w konfiguracji z dyskiem sztywnym. Producent narzuca jako minimum dwa napędy dyskietek oraz 320 KB pamięci RAM.

Treść ekranu bardzo przypomina menu starej wersji WordStara. Najwyższy wiersz służy do wyświetlania statusów: nazwy dokumentu, numeru aktualnej strony, położenia kursora i aktualnego trybu wprowadzania. Niżej znajduje się dobrze uporządkowane i przejrzyste menu, zaś pod nim – wiersz formatowania zawierający znaczniki marginesów i punkty tabulacji.

Użytkownik decyduje sam o łącznym założonym rozmiarze potrzebnych informacji. Początkujący każą sobie wyświetlać wszystkie rodzaje menu. W tym wypadku informacje pomocnicze zajmują aż pół ekranu. Średnio zaawansowani mogą oglądać tylko menu podrzędne. Doświadczeni wybierają trzecią alternatywę, tzn. pracę bez menu; informacje pomocnicze zajmują tylko dwa górne wiersze, zawierające informacje o stanie i formacie. Na wyświetlanie tekstu pozostają 22 wiersze.

WordStar 2000 dostarczany jest w kilku wersjach językowych, m.in. w angielskiej i niemieckiej.

Wprowadzanie poleceń bez użycia klawiszy funkcyjnych przebiega dwustopniowo. Najpierw wciska się klawisz CTRL i literę menu podrzędnego, a potem literę polecenia w tym menu. Często może to być nieco kłopotliwe.

Inny sposób wprowadzania polega na użyciu klawiszy funkcyjnych, które trzeba wcisnąć tylko raz. Klawisze te trudno jest jednak powiązać mnemotechnicznie z wykonywanymi poleceniami.

Funkcje spełniane przez poszczególne klawisze funkcyjne można przedefiniować.

FORMATOWANIE

Słynne ze starej wersji programu WordStar *polecenia z kropką na początku* należą już do przeszłości. W wersji 2000 wygląd tekstu pojawiającego się na ekranie zostaje dokładnie zdefiniowany jeszcze przed rozpoczęciem wprowadzania: każdemu rozpoczynanemu dokumentowi przyporządkowuje się zawsze opis formatowania. Na dyskietce znajdują się pewne standardowe wzory formatowania dokumentów, inne – w zależności od potrzeby – można zdefiniować i zapamiętać na tym nośniku.

Definiowanie sposobu formatowania trudno byłoby chyba zorganizować prościej. Po otwarciu nowego dokumentu następuje wprowadzanie w sposób konwersacyjny odpowiednich parametrów. Program proponuje pewne wartości standardowe. Wciśnięcie samego klawisza RETURN oznacza zgodę na przyjęcie wartości standardowej. Aby program przyjął inną wartość, trzeba ją wprowadzić z klawiatury. Także i tutaj program oferuje pomoc w postaci czterech stron komentarzy opisujących dokładnie każde pytanie.

Wiele parametrów (jak podkreślenia, wy-

tłuszczenia, lewy i prawy margines czy tabulatory) można później zmieniać w tekście. Polecenia wprowadza się bez ich uwidaczniania. W przeciwieństwie do wielu innych programów przetwarzania tekstów, liczba wprowadzonych poleceń tego typu nie jest ograniczona z góry.

SZYBKOŚĆ PRACY

Wersja 1.0 wymaga jeszcze pewnego dopracowania, także pod względem szybkości. Wydaje się, że WordStar 2000 nie najlepiej pracuje na komputerach z mikroprocesorem 8088 i bez sztywnego dysku.

Do wolniejszego tempa pracy przyczynia się sposób formatowania. Po skasowaniu fragmentu tekstu WordStar rozpoczyna przepisywanie całości. Przy obszernych przestawieniach tekstu (np. przesunięcie treści stron 3 i 4 i wstawienie jej po stronie 16) zajmuje to czas wystarczający na wypicie filiżanki kawy. Inne programy – wśród nich także poprzedni WordStar – czekają z zapisem do momentu otrzymania polecenia uporządkowania ekranu. Mimo takiego sposobu działania programu większość użytkowników ocenia go pozytywnie. Jego zalety widać np. przy przesuwaniu kolumn liczbowych – wystarczy do tego kilka przestawień tabulatorów.

Jeśli komputer jest wyposażony w kartę grafiki, to na ekranie można uzyskiwać wytłuszczoną czcionkę, podkreślenia oraz umieszczać znaki w położeniu górnych i dolnych indeksów. Bez karty graficznej wyróżnienia te są sygnalizowane za pomocą rozjaśnienia lub inwersji, a na monitorach barwnych – za pomocą innego koloru. Skala barw wybrana przez MicroPro opiera się na pastelowych tonach z lat pięćdziesiątych, można ją jednak zmieniać przy instalowaniu programu.

OPCJE

Liczne opcje, nie istniejące w starej wersji lub osiągalne tylko po zakupieniu dodatkowego oprogramowania, zawarto teraz w podstawowym programie WordStar 2000. Dzięki temu w procesie pisania tekstu można korzystać z pewnych dodatkowych możliwości. Obecność większości z nich w bardzo dobrych programach tego typu jest obecnie wymaganiem podstawowym (np. wielka różnorodność wierszy z nagłówkiem lub stopką).

Firma MicroPro długo ociągała się z wprowadzeniem techniki okien. Obecnie obszar ekranu można podzielić na dwie lub trzy równe części. Nie jest to wiele w porównaniu z produktem firmy Microsoft, który daje do dyspozycji osiem okien, a ponadto ich wielkość można dowolnie zmieniać. Chociaż trzeba przyznać, że racjonalna praca jest możliwa najwyżej z dwoma lub trzema oknami.

Najlepsze programy przetwarzania tekstów, takie jak MS Word, Word Perfect i Samna Word, organizują też tzw. stopki, czyli wiersze na dole strony, poniżej właściwego tekstu. Ich wygląd może być oczywiście różny. Na pozycji, na której ma wystąpić stopka, wprowadza się odpowiednie polecenie. Za pomocą jednego lub dwóch znaczników definiuje się obszar, w którym ma wystąpić dopisek. Można go drukować na każdej stronie lub tylko na końcu tekstu. W odróżnieniu od MS Word, WordStar 2000 nie umożliwia jeszcze definiowania tekstu stopki bezpośrednio na danej stronie.

MAILMERGE

Mailmerge, czyli nakładkę programową wymienianą łącznie z nazwą WordStar, można było dotąd nabyć jedynie za dodatkową opłatą. Obecnie stanowi ona wbudowaną na stałe część programu WordStar 2000. Przydatna jest przy znacznej liczbie podobnie brzmiących listów, różniących się jedynie zmiennymi polami, takimi jak np. adres, zwroty grzecznościowe czy wykaz wyrobów. Również tu zrezygnowano z rozkazów zaczynających się od kropki.

Wsród opcji znajdują się też i takie, które nie odgrywają większej roli przy podejmowaniu decyzji zakupu, mają jednak pewne znaczenie użytkowe. Przykładem może być użyteczny przy opracowywaniu rękopisów tzw. komentarz wewnętrzny, który pojawia się na ekranie, lecz który nie jest potem drukowany.

Inną opcją są tzw. moduły stenograficzne. Wielokrotnie używanym fragmentom tekstu (o maksymalnej długości 560 znaków) można nadać odpowiednie nazwy i zapamiętać je na dyskietce. Wystarczy później napisać taką nazwę i wcisnąć klawisz ESC, a cały zapamiętany fragment zostanie wprowadzony do tworzonego dokumentu. Oczywiście należy rozsądnie gospodarować zapamiętywanymi tekstami. Jedna dyskietka pozwala na zapamiętanie do 20 takich fragmentów po 2000 znaków każdy.

Po pewnym czasie użytkownik doceni możliwość zapamiętywania także poleceń redagujących. Jest to szczególnie wygodne przy przetwarzaniu tekstów zawierających wiele wyróżnień (np. drukowaną wytłuszczoną czcionką nazwę firmy).

W programie WordStar 2000 istnieje możliwość wykonywania obliczeń oraz sortowania w porządku alfabetycznym lub liczbowym. Opcja ta wchodzi w skład *operacji blokowych*.

Tabele, kolumny liczb, rejestry haseł czy wykazy literatury należy najpierw oznaczyc jako bloki. Następnie można rozpocząć sortowanie. Przetworzony fragment tekstu (składający się najwyżej z 50 wierszy) można uporządkować kolumnami lub wierszami, w porządku wzrastającym lub malejącym według 20 początkowych znaków.

W programie WordStar istnieje możliwość wykonywania czterech podstawowych działań potęgowania. Wystarczy wyróżnić blok liczb, wprowadzić symbol operacji arytmetycznej i podać polecenie *obliczaj*. Niestety nie zaimplementowano obliczania procentów. Kłopotliwe jest kasowanie symbolu operacji (np. przy mnożeniu) po każdorazowym przeprowadzeniu obliczeń.

DRUKARKI

Użytkownik ma możliwość użycia jednego ze 119 modeli drukarek. Jeśli jednak dysponuje drukarką nie mieszczącą się w tym zbiorze, łatwo może zdefiniować jej cechy i włączyć ją do realizacji programu. Szkoda tylko, że program umożliwia równoczesne korzystanie tylko z jednej drukarki. Jeżeli zamierzamy wprowadzać na dokumenty krótkie teksty, np. zaadresowanie koperty lub wypełnienie formularza, to WordStar 2000 może być używany w trybie bezpośredniego drukowania, czyniąc z komputera i drukarki zwykłą maszynę do pisania.

* * *

WordStar 2000 firmy MicroPro nie jest jakąś nadzwyczajną nowością. Nie ma w nim niczego, czego nie można by znaleźć w produktach innych firm. Nie powstał również żaden nowy standard. Jednak połączenie wszystkich funkcji powoduje, że WordStar 2000 jest doskonałym programem, który oprócz dużej wydajności jest również bardzo wygodny dla użytkownika. Sloganu reklamowego: *przetwarzanie tekstów dla każdego* nie należy jednak brać zbyt dosłownie. Koszty i wysiłek włożone w naukę posługiwania się programem opłacą się tylko wtedy, gdy rzeczywiście będzie wykorzystany bogaty zestaw funkcji oferowanych przez program.

micro

Przepraszamy za pominięcie w stopce zeszytu 3 „Mikroklanu" nazwiska red. KRZYSZTOFA KONTKA.

BASIC DLA POCZĄTKUJĄCYCH

W poprzednim odcinku zajmowaliśmy się szczegółowo pętlami FOR...NEXT. Tekst w ramce zawiera zestawienie najważniejszych wiadomości z poprzednich odcinków. Obecnie przedstawiamy sposób posługiwania się podprogramami oraz ładowanie i zapamiętywanie programów w pamięci zewnętrznej.

Jak zapewne pamiętasz, w poprzednim odcinku zaproponowaliśmy napisanie programu, który obliczałby odsetki uzyskiwane z kapitału o wartości od 1000 do 20 000 dolarów przy oprocentowaniu od 5% do 10% oraz okresie od 10 do 20 lat. Rozwiązanie zadania znajduje się na wydruku 1. Jak widać, program składa się niemal wyłącznie z pętli FOR...NEXT. Oczywiście istnieją też inne rozwiązania tego problemu, niemniej użycie pętli programowych jest tu naturalne i efektywne.

W programie pokazanym na wydruku 1 opisy wyprowadzanych danych są drukowane przy każdym przejściu przez wewnętrzną pętlę (wiersz 50). Można wyprowadzić je przed wydrukowaniem całego bloku danych – wtedy wiersz o numerze 50 powinien otrzymać numer 15 albo 25. Aby wyniki nie znikały zbyt szybko z ekranu, wprowadzono pętle opóźnienia (wiersze 70 i 80)

Zmień program tak, aby wykonywał odpowiednie obliczenia dla wartości kapitału od 50 do 500 dolarów, narastających w 50 krokach, dla oprocentowania równego 5,75 oraz okresów od 1 do 12 miesięcy. Drukuj opisy wyników dla każdej nowej wartości kapitału. Rozwiązanie jest przedstawione na wydruku 2.

ZAOKRĄGLENIE

Po wykonaniu programu możesz się przekonać, że wyniki są drukowane w postaci liczb z najwyżej ośmioma cyframi po przecinku dziesiętnym[1]. Ponieważ odsetki podawane są zawsze w postaci odpowiednich kart dolarów i centów, otrzymane wyniki należy zaokrąglić. Pomocą w tym wypadku może służyć tzw. funkcja INTEGER, która liczbie z przecinkiem dziesiętnym, nazywanej liczbą typu REAL lub liczbą zmiennoprzecinkową, przyporządkowuje liczbę całkowitą (zwaną liczbą typu INTEGER). Wszystkie cyfry po przecinku są odrzucane.

Wróćmy na chwilę do trybu bezpośredniego. Funkcja INTEGER jest wywoływana podobnie jak np. funkcja pierwiastkowania, tyle że zamiast symbolu SQR występuje tu symbol INT:

INT (liczba)

W miejscu liczby może występować wyrażenie arytmetyczne.

Zapamiętaj! Za pomocą funkcji INT ( ) można odrzucić cyfry dziesiętne znajdujące się po przecinku. W wyniku otrzymuje się liczbę całkowitą.

Wprowadź w trybie bezpośrednim:

?INT (4)

Po wciśnięciu klawisza *RETURN* na ekranie pojawi się również 4. Argument funkcji był od razu liczbą całkowitą, dlatego wartość nie musiała być zaokrąglana.

Zapamiętaj! Liczby całkowite nie są zaokrąglane.

Wprowadź teraz:

?INT (4.5)

i wciśnij ponownie klawisz *RETURN*.W wyniku pojawi się znów liczba 4. Znaki .5 zostały odrzucone:

?INT (3.3+3.3)

Wynikiem jest 6.

Zapamiętaj! W nawiasach funkcji INT mogą znajdować się wyrażenia arytmetyczne, składające się z liczb lub nazw zmiennych oraz operatorów arytmetycznych.

Wprowadź:

?INT (3.3)+(3.3)

Wynikiem znowu jest 6

Wprowadź jednak:

?INT (3,4+6,7)

Wynikiem jest 10, gdyż najpierw następuje wykonanie operacji wewnątrz nawiasów (3.4+6.7 = 10.1), a dopiero potem odrzucenie przecinka i znajdujących się po nim cyfr.

Gdy wprowadzisz:

?INT (3.4)+INT(6.7)

Wynikiem będzie 9, gdyż w każdej liczbie najpierw nastąpi odrzucenie części dziesiętnych, a dopiero potem dodanie obu składników.

Wprowadź:

?INT (–3.4)

Wynikiem jest –4. Dlaczego? Otóż fnkcja INT oblicza najbliższą liczbę całkowitą mniejszą od argumentu lub równą temu argumentowi. W wypadku liczb dodatnich sprowadza się to do odrzucenia liczb znajdujących się po przecinku. W wypadku liczb ujemnych może się też zmienic cyfra znajdująca się bezpośrednio przed przecinkiem. W ostatnim przykładzie najbliższa liczba całkowita mniejsza lub równa liczbie –3.4 jest właśnie liczba –4.

Zapamiętaj! Jeśli argumentem funcji INT jest liczba ujemna, to w wyniku otrzymuje się najbliższą liczbę całkowitą mniejszą lub równą argumentowi.

Aby więc zaokrąglić daną liczbę do najbliższej liczby całkowitej, nie wystarczy użyć funkcji INT. Trzeba dodac do argumentu liczbę 0.5. Wtedy liczby dodatnie w wartości części ułamkowej większej od 0.5 zostaną

[1] W podanych przykładach występuje kropka dziesiętna, zgodnie z zasadą stosowaną w zapisie angielskim

```
10 FOR KA=1000 TO 20000 STEP 1000
20 FOR PR=5 TO 10
30 FOR ZE=10 TO 20
40 ZI=KA*PR*ZE/100
50 PRINT"KAPITAL","PROCENT","LATA","ODSETKI"
60 PRINT KA,PR,ZE,ZI
70 FORWA=1TO1000
80 NEXT
90 NEXT
100 NEXT
110 NEXT
```

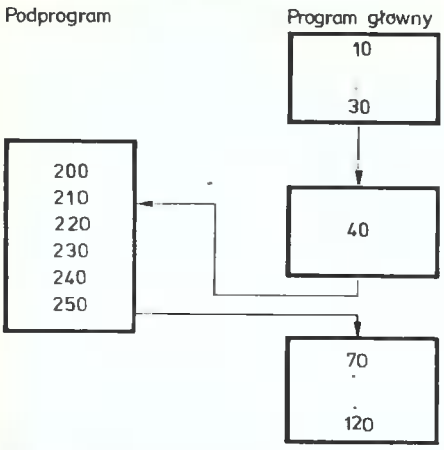
Wydruk 1. Rozwiązanie poprzedniego zadania

```
10 FOR KA=50 TO 500 STEP 50
15 PRINT "KAPITAL","PROCENT","LATA","ODSETKI"
20 PR=5.75
30 FOR ZE=1/12 TO 1 STEP 1/12
40 ZI=KA*PR*ZE/100
60 PRINT KA,PR,ZE,ZI
70 FORWA=1TO1000
90 NEXT
100 NEXT
110 NEXT
120 END
```

Wydruk 2. Inne możliwe rozwiązanie

```
60 PRINT INT(KA*100+0.5)/100,
61 PRINT INT(PR*100+0.5)/100,
62 PRINT INT(ZE*100+0.5)/100,
63 PRINT INT(ZI*100+0.5)/100
```

Wydruk 3. Zmiana sposobu zaokrąglania



Rys. 1. Schemat przebiegu programu przy wywoływaniu podprogramów

zaokrąglone w górę, natomiast liczby dodatnie o wartości części ułamkowej mniejszej od 0.5 – w dół. Liczby ujemne będą także zaokrąglane do najbliższej wartości całkowitej. Możesz to sprawdzić, wprowadzając:

?INT (3.4+0.5)
?INT (3.6+0.5)
?INT (–3.4+0.5)
?INT(–3.6+0.5)

Taki sposób zaokrąglania będziemy wykorzystywać w następnych programach. Najpierw jednak omówimy jeszcze jedną funkcję. Służy ona do obliczania wartości bezwzględnej argumentu.

Wprowadź:

?ABS (3.4)

Na ekranie pojawia się jako wynik liczba równa argumentowi, tzn. 3.4.

Wprowadź teraz:

?ABS (–3.4)

Znów wynikiem jest 3.4. Wyeliminowany został jednak znak liczby.

Co zrobić, aby wydrukować cenę w postaci liczby z dokładnie dwiema cyframi po przecinku? Takie przedstawienie potrzebne jest najczęściej. Za pomocą samej funkcji INT nie osiągnie się tego celu. Trzeba wykonać kilka operacji:

- zamienić dolary i centy na centy,
- zaokrąglić liczbę centów do najbliższej liczby całkowitej,
- zamienić ponownie centy na dolary (z częścią ułamkową).

Przypuśćmy, że chcemy w ten sposob zaokrąglić liczbę 3.333. Najpierw mnożymy ją przez 100 i otrzymujemy liczbę centów:333.3. W celu prawidłowego zaokrąglenia (do najbliższej wartości całkowitej) dodajemy 0.5 i dla takiego argumentu obliczamy wartość funkcji INT otrzymując 333. Wynik znowu przeliczamy na dolary, dzieląc przez 100 i otrzymujemy w rezultacie 3.33.

Wszystkie te operacje można przeprowadzić za pomocą jednego polecenia:

?INT (3.333*100+0.5)/100

Zastąp teraz 3.333 przez 6.666 i sprawdź rezultat.

Opisaną funkcję wprowadzimy teraz do naszego programu. Wiersz 60 ulegnie rozszczepieniu na cztery wiersze zawierające poszczególne wyniki (wydruk 3).

Zwróć jeszcze raz uwagę na użycie znaku przecinka (,) na końcu wierszy 60, 61 i 62. Powoduje on brak przesuwu do nowego wiersza, po wydrukowaniu ostatniej liczby. Dzięki temu wszystkie cztery wyniki drukowane są w tym samym wierszu. Po ostatnim wyniku (drukowanym przez polecenia w wierszu 63) chcemy przesunąć się do nowego wiersza – dlatego w wierszu 63 nie ma na końcu przecinka.

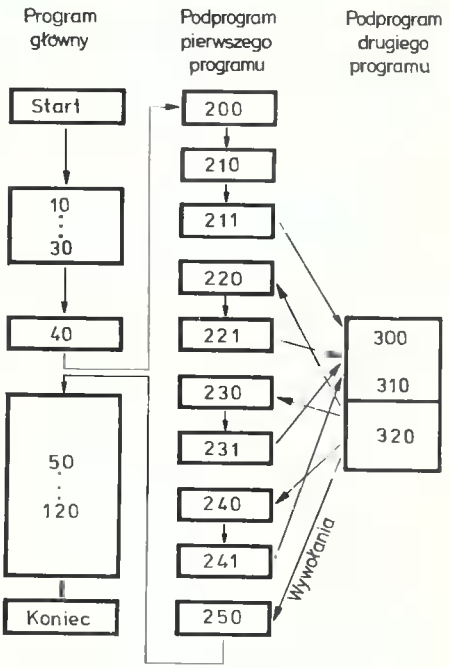
Jeśli obejrzysz wyniki działania programu, przekonasz się, że nie wszystko jest jeszcze w porządku. Otóż jeżeli liczba kończy się jednym lub dwoma zerami po przecinku dziesiętnym, to nie są one drukowane. Wynika to z właściwości komputera. W dalszej części kursu pokażemy, jak zlikwidować tę ostatnią niedogodność.

PODPROGRAMY

Przedstawiona sekwencja prowadząca do prawidłowego zaokrąglenia może być używana w dużych programach wielokrotnie. Czy trzeba też wielokrotnie wprowadzać zestaw odpowiednich wierszy programu? Otóż niekoniecznie. Można je wprowadzić tylko w jednym miejscu, a potem w wielu miejscach programu nakazywac ich wykonanie. Na tym polega sens istnienia podprogramów. Wywołanie podprogramu jest nieco podobne do wykonania polecenia skoku *GOTO:*

GOSUB nr wiersza

GOSUB jest skrótem od angielskich słów, GO to SUBroutine (skocz do podprogramu).



Rys. 2. Schemat przebiegu programu z wydruku 5 – uwidocznione poszczególne poziomy wywołań podprogramów

Po wykonaniu instrukcji składających się na podprogram trzeba wrócić do miejsca, z którego został on wywołany. Nie da się tego zrealizować za pomocą polecenia *GOTO*, znajdującego się na końcu podprogramu, gdyż w poleceniu tym musi występować jeden konkretny adres, a idea podprogramu polega na wywołaniu go z różnych miejsc w programie. Do powrotu z podprogramu służy inne polecenie – *RETURN* (ang. wróć). Powoduje ono powrót do polecenia znajdującego się bezpośrednio za poleceniem *GOSUB*, powodującym wywołanie podprogramu. Na rys. 1 znajduje się ilustracja graficzna przebiegu wykonania kolejnych poleceń przy wywołaniu podprogramu.

Zapamiętaj! Podprogramy są fragmentami programu, wywoływanymi za pomocą polecenia GOSUB nr wiersza, z których wyjście następuje po wykonaniu polecenia RETURN.

```
10 FOR KA=50 TO 500 STEP 50
15 PRINT "KAPITAL","PROCENT","LATA","ODSETKI"
20 PR=5.75
30 FOR ZE=1/12 TO 1 STEP 1/12
40 GOSUB200
70 FORWA=1TO1000
90 NEXT
110 NEXT
120 END
200 ZI=KA*PR*ZE/100
210 PRINT INT(KA*100+0.5)/100
220 PRINT INT(PR*100+0.5)/100
230 PRINT INT(ZE*100+0.5)/100
240 PRINT INT(ZI*100+0.5)/100
250 RETURN
```

Wydruk 4. Wiersze programowe 60-63 z wydruku 3 w postaci podprogramu (210-250)

```
210 X=KA
211 GOSUB300
212 PRINT Y,
220 X=PR
221 GOSUB300
223 PRINT Y,
230 X=ZE
231 GOSUB300
232 PRINT Y,
240 X=ZI
241 GOSUB300
242 PRINT Y
```

Wydruk 5. Ta sekwencja zastępuje w programie głównym poprzednie rozkazy wyprowadzające informacje na ekran

```
10 FOR KA=50 TO 500 STEP 50
15 PRINT "KAPITAL","PROCENT","LATA","ODSETKI"
20 PR=5.75
30 FOR ZE=1/12 TO 1 STEP 1/12
40 GOSUB200
50 PRINT
70 FORWA=1TO1000
90 NEXT
100 NEXT
110 NEXT
120 END
200 ZI=KA*PR*ZE/100
210 X=KA
211 GOSUB300
220 X=PR
221 GOSUB300
230 X=ZE
231 GOSUB300
240 X=ZI
241 GOSUB300
250 RETURN
300 Y=INT(X*100+0.5)/100
310 PRINT Y,
320 RETURN
```

Wydruk 6. Kompletny listing zmienionego programu

Najlepiej zademonstrować to na konkretnym przykładzie. W tym celu wiersze 40 do 63 przedstawimy w postaci podprogramu. Zamień program w sposób pokazany na wydruku 4.

Program jest wykonywany w ten sposób, że po każdym osiągnięciu wiersza 40 następuje skok do wiersza 200. Komputer zapamiętuje jednoczesnie, że skok był właśnie z wiersza 40. Dzięki temu po napotkaniu polecenia *RETURN* wiadomo, dokąd trzeba wrócić.

PRZEKAZYWANIE PARAMETRÓW DO PODPROGRAMÓW

Przedstawiony podprogram jest przykładem podprogramu pobierającego tzw. parametry z programu głównego. Parametrami są w tym wypadku zmienne *KA*, *PR* i *ZE*.

Zapamiętaj! Zmienne w podprogramie mają na początku tę samą wartość, którą miały w chwili wywołania podprogramu przez program główny.

Podprogramy mogą też mieć parametry wejściowe, czyli zmienne, których wartości są obliczane przez podprogram i przekazywane z powrotem do programu nadrzędnego.

Zapamiętaj! Za pomocą tzw. parametrów wyjściowych (albo zwrotnych) można przekazywać wartości z podprogramu z powrotem do programu nadrzędnego.

Jak widać, parametry nie są niczym szczególnym dla podprogramu – to po prostu pewne zmienne. Jednak ze względu na prawidłowe udokumentowanie programu należy ustalić, jakich zmiennych używa dany podprogram. Przy złożonych programach bardzo łatwo mogą pojawić się błędy wynikające z naprzemiennego używania tych samych zmiennych przez różne podprogramy.

Wprowadź teraz następujące dodatkowe wiersze:

```
300 X = INT(Y*100+0.5)/100
310 RETURN
```

Napisaliśmy w ten sposób podprogram, który może być wykorzystany do obliczenia wartości zmiennych *KA*, *PR*, *ZE* i *ZI* przez przypisanie im wartości zmiennej *X*. Wiersz 300 mógłby też mieć nieco inną postać:

```
300 X = INT(X*100+0.5)/100
```

W postaci tej *X* jest jednocześnie parametrem wejściowym i wyjściowym.

Zapamiętaj! Parametry wejściowe mogą być jednocześnie parametrami wyjściowymi.

Zastąpimy teraz polecenia drukujące wyniki w programie głównym sekwencją poleceń przedstawioną na wydruku 5.

Jeśli jesteś dociekliwy, być może powiesz teraz: dlaczego nie usunąć wierszy 212, 222, 232 i 242, a zamiast nich umieścić polecenie drukowania w podprogramie?

Jednym z możliwych rozwiązań jest program z wydruku 6. Szczególnie interesujący jest tu wiersz 50, który w każdym przebiegu pętli przed obliczeniem wszystkich wyników powoduje wydrukowanie pustego wiersza (za pomocą polecenia PRINT bez parametrów).

POWTÓRZENIE WIADOMOŚCI Z POPRZEDNICH ODCINKÓW

★ Wprowadzanie kończy się wciśnięciem klawisza RETURN.
★ Po poleceniu INPUT można umieścić tekst ograniczony z obu stron znakiem cudzysłowu, opisujący bliżej wprowadzoną zmienną. Tekst ten musi być koniecznie oddzielony znakiem średnika od występującej po nim zmiennej.
★ Do wyprowadzania tekstów w poleceniu INPUT nie można używać zmiennych łańcuchowych (zmiennych tekstowych).
★ W poleceniu PRINT mogą występować razem zmienne, stałe, łańcuchy (ciągi) znaków oraz liczby.
★ Za pomocą polecenia GOTO mozna zmieniać normalny przebieg programu. Za poleceniem GOTO musi się znajdować numer wiersza, do ktorego ma nastąpić skok.
★ Pętle programowe tworzy się za pomocą poleceń FOR...NEXT.
★ Pętle mogą być w sobie zagnieżdżone. Dla każdej pętli trzeba używać oddzielnej zmiennej sterującej. Wartości początkowe i końcowe mogą być jednak tymi samymi zmiennymi lub liczbami.
★ Zmienne sterujące po wyjściu z pętli mają wartość równą wartości końcowej pętli zwiększonej o 1.
★ Wartości zmiennych można obejrzeć w trybie bezpośrednim za pomocą zwykłych poleceń PRINT.
★ Interpreter programu BASIC pozwala w każdej chwili na zatrzymanie programu i sprawdzenie wartości zmiennych.
★ W opisie pętli można zdefiniować przyrost wartości zmiennej sterującej. W tym celu po wartości końcowej za słowem STEP wprowadza się odpowiednią wartość – wielkość kroku, która przy każdym przejściu przez pętlę jest dodawana do wartości zmiennej sterującej.
★ Jeśli nie wprowadzi się wielkości kroku, jest ona automatycznie przyjmowana za równą 1.
★ Wielkością kroku może być także liczba niecałkowita (z przecinkiem dziesiętnym, np. 1.5).

W naszym przykładzie w czasie przebiegu programu na ekranie pojawia się jeszcze jeden dodatkowy wiersz pusty. Spowodowane jest to specyfiką obrazowania informacji na ekranie mikrokomputera Commodore. Spróbuj wykonać jeszcze inną wersję tego programu, w której drukowane będą tylko kapitał, czas i odsetki (bez wielkości oprocentowania).

Być może zauważyłeś, że jeden podprogram wywołuje inny podprogram.

Zapamiętaj! Podprogram może być wywoływany z innego podprogramu.

Podprogram rozpoczynający się w wierszu 300 jest czterokrotnie wywoływany przez podprogram rozpoczynający się w wierszu 200. Takie zagnieżdżanie jest właściwie racją istnienia podprogramów. Na rys. 2 pokazano schemat blokowy programu.

ŁADOWANIE I ZAPAMIĘTYWANIE PROGRAMÓW

Niewątpliwie każdorazowe wprowadzanie z klawiatury przedstawionego programu byłoby bardzo nużące. Dlatego w języku BASIC istnieją dwa polecenia umożliwiające ładowanie i zapamiętywanie używanych programów. Należy jedynie nadać programowi nazwę, którą w wypadku mikrokomputera Commodore 64 może składać się najwyżej z szesnastu liter i cyfr. Program nasz nazwiemy ODSETKI.

Uwagi dla użytkowników ZX Spectrum

ZX Spectrum bez dodatkowego interfejsu umożliwia współpracę jedynie z magnetofonem. W celu zapisania zbioru na taśmie należy wprowadzić polecenie

***SAVE* nazwa *LINE* nr wiersza**

Nazwa jest zmienną łańcuchową (zmienną tekstową) lub łańcuchem znaków (ciągiem znaków). Nazwa może być maksymalnie dziesięcioliterowa. Człon *„LINE nr wiersza"* jest opcjonalny. Jeśli występuje, to po załadowaniu programu z taśmy do pamięci komputera rozpocznie się jego automatyczne wykonywanie od podanego numeru wiersza. W języku BASIC do ładowania programu do pamięci komputera służy polecenie

***LOAD* nazwa**

Nazwa może być zmienną łańcuchową (zmienną tekstową) lub łańcuchem znaków (ciągiem znaków). Jeśli jest to ciąg pusty, to zostanie załadowany pierwszy napotkany na taśmie zbiór. Jeśli nie, komputer pominie zbiory z innych nazwach i dopiero po napotkaniu zbioru o podanej nazwie załaduje go do pamięci. Przy zapisywaniu zbioru na taśmie, przy użyciu opcji *LINE*, program natychmiast po załadowaniu rozpocznie działanie.

W wypadku komputera Commodore 64 ważne jest też urządzenie służące do zapamiętywania programów (magnetofon kasetowy lub stacja dysków elastycznych). W poleceniu służącym do zapamiętywania programu na nośniku zewnętrznym (SAVE) oraz w poleceniu służącym do ładowania programu z nośnika zewnętrznego do pamięci komputera (LOAD) trzeba podać po przecinku numer urządzenia. Magnetofon kasetowy ma numer 1, natomiast stacja dysków elastycznych – numer 8.

Tak więc do zapamiętania naszego programu na dyskietce służy polecenie:

SAVE "ODSETKI", 8

zaś do zapamiętania programu na taśmie kasetowej polecenie:

SAVE "ODSETKI", 1

Przy korzystaniu z dyskietek należy zwrócić uwagę czy przypadkiem nie mruga czerwona lampka. Jeśli tak, to znaczy, że przy zapisie nastąpił błąd i należy powtórnie zapamiętać program. Przy korzystaniu z magnatofonu, w celu sprawdzenia poprawności zapisu, należy cofnąć taśmę przed miejsce początku programu i wprowadzić z klawiatury polecenie:

VERIFY

Teraz trzeba nacisnąć jeszcze przycisk *PLAY* w magnetofonie. Komputer czyta program z taśmy i porównuje go z wersją wciąż jeszcze istniejącą w jego pamięci wewnętrznej.

Zapamiętywanie programu polega na kopiowaniu go z pamięci wewnętrznej na nośnik zewnętrzny (dyskietkę albo kasetę magnetofonową). Program w pamięci wewnętrznej nie ulega przy tym zniszczeniu.

Zapamiętane na nośniku zewnętrznym programy można ponownie załadować do pamięci. Przy odczycie z dyskietki odpowiednie polecenie będzie miało postać:

LOAD "ODSETKI", 8

natomiast przy odczycie z kasety magnetofonowej:

LOAD "ODSETKI", 1

Oczywiście program zapisany na taśmie czy dyskietce nie ulega przy tej operacji zniszczeniu.

Na zakończenie proponujemy Ci rozwiązanie następującego problemu: należy napisać program obliczający długość obwodu i przekątnej oraz pole powierzchni prostokątów o długościach od 1 do 20 i szerokościach od 1 do 20. Wskazówka: przy wprowadzaniu danych i wyników zamiast przecinka (,) używaj znaku średnika (;), gdyż pozwoli to wyprowadzić długość obu boków oraz trzy znalezione wartości w tym samym wierszu. Do obliczeń używaj podprogramu, który będzie wyprowadzał także wyniki. Użyj funkcji SQR() do obliczenia długości przekątnej. Jeśli nie pamiętasz potrzebnych wzorów, zajrzyj do podręcznika szkolnego albo poradnika matematycznego.

micro

CZYM JEST UNIX?

POWŁOKA

Powłoka Unixa (ang. shell), przedstawiona w pierwszym odcinku (na str. 20 pierwszego zeszytu Mikroklanu) jako pierścień otaczający jądro, jest interpreterem poleceń kierowanych do systemu operacyjnego, a więc takim samym programem jak inne programy usługowe, lecz spełniającym wyspecjalizowaną funkcję. Zgodnie ze swą nazwą powłoka otacza jądro systemu operacyjnego i izoluje je od otoczenia. Jest inicjowana automatycznie po zarejestrowaniu się użytkownika w systemie (ang. login). Wszystkie następne polecenia do systemu operacyjnego są kierowane do niego przez powłokę.

Ogólna postać polecenia jest następująca:

polecenie argument_1 argument_2 ...

Argumenty mogą być trzech rodzajów – pliki, opcje i pozostałe – zależnie od rodzaju polecenia. Opcje dotyczą sposobu wykonania polecenia i są zawsze jednoliterowe (choć można łączyć litery w grupy), poprzedzone myślnikiem.

Powłoka przyjmuje polecenie od użytkownika, dekoduje je i przekazuje do jądra w celu wykonania. Pracę powłoki można przedstawić algorytmicznie jako ciąg następujących czynności:

- wysłanie znaku gotowości (np. $) na terminal,
- przyjęcie polecenia od operatora (lub z pliku),
- zdekodowanie polecenia i odpowiadającego mu programu,
- przekazanie polecenia do jądra w celu wykonania (dokładniej, wykonanie polecenia polega na utworzeniu odpowiedniego procesu – lub procesów – i oczekiwaniu na jego zakończenie),
- przyjęcie odpowiedzi od jądra, np. wyników wykonanego polecenia lub stanu,
- powrót do punktu pierwszego.

Istnieje jednak więcej cech wyróżniających powłokę od innych programów usługowych. Są to, przede wszystkim, mechanizmy umożliwiające wzajemną współpracę programów usługowych, tj. ograniczoną współbieżność w wykonywaniu poleceń. Umożliwiają one także programowanie powłoki.

MECHANIZMY WSPÓŁPRACY PROGRAMÓW

Powłoka Unixa umożliwia znacznie więcej niż wydanie polecenia systemowi operacyjnemu i oczekiwanie na pomyślne zakończenie jego wykonania, jak wynika z ciągu czynności wyliczonych w poprzednim punkcie. Dla użytkownika szczególnie istotna jest możliwość łączenia poleceń (tj. współpracy programów) oraz ich parametryzowania (lub wręcz – zmiany funkcji bez parametryzowania).

Do najprostszych, lecz bardzo skutecznych mechanizmów umożliwiających tego rodzaju pracę, z których część zastosowano po raz pierwszy w Unixie, należą:

- przeadresowywanie wejścia-wyjścia (ang. I/O redirection)
- tworzenie potoków (ang. pipes)
- filtrowanie.

Przeadresowywanie wejścia-wyjścia polega na zmianie standardowych urządzeń wprowadzania i wyprowadzania informacji – są nimi zwykle klawiatura i ekran terminala – na inne wygodne dla użytkownika. Kierunki wprowadzania i wyprowadzania są oznaczane odpowiednio znakami "<" i ">". Przykładowo, używając polecenia *ls*, znanego z drugiego odcinka tego cyklu, można wyprowadzić wykaz plików zawartych w skorowidzu bieżącym na dowolny plik (w tym wypadku nazwany file) zamiast na ekran terminala:

$ls – l > file

Tekst "> file" będący częścią polecenia jest interpretowany przez powłokę i nie przekazywany do programu *ls*. Przeadresowania dokonuje więc sama powłoka. Informacja wyprowadzana jest zapisywana na plik *file* (jego poprzednia zawartość ulega zniszczeniu) po jego uprzednim utworzeniu (gdy jeszcze nie istnieje). Jeśli nie należy niszczyć poprzedniej zawartości pliku, to można do niego dołączyć wyprowadzaną informację używając znaków ">>".

Tą samą metodą można wyprowadzać jednocześnie informacje dostarczane przez kilka poleceń, podając ich nazwy rozdzielone średnikami w jednej linii i ujmując w nawiasy, np.:

$ (date; ls – l) >> file

Polecenie *date* powoduje wyprowadzenie bieżącej daty i aktualnego czasu. Jeżeli w powyższym poleceniu złożonym zabrakłoby nawiasów, to byłoby ono również poprawne, z tą różnicą że polecenie *date* spowodowałoby wyprowadzenie informacji na ekran terminala, a polecenie *ls* – na plik *file*.

Podobnie jak zmienia się standardowy plik wyjściowy, można również zmienić standardowy plik wejściowy (standardowe wejście) posługując się znakiem "<". Interesującym wariantem przeadresowywania wejścia-wyjścia jest wprowadzanie informacji z tzw. plików miejscowych (ang. here documents),

oznaczane znakami "<<", polegające na przechowywaniu danych dla poleceń razem z samymi poleceniami.

Standardowe wejścia i wyjścia można łączyć wzajemnie tworząc potoki, np. ciąg poleceń:

$ls -l > file
$wc < file

można zastąpić jednym – potokiem

$ls | wc

Specjalny rodzaj potoków przeznaczonych do wyekstrahowania określonej informacji z pliku lub urządzenia nazywa się filtrami.

Do podstawowych funkcji zalicza się jeszcze możliwość przetwarzania drugoplanowego, symbolizowaną przez zakończenie linii polecenia znakiem &, np.:

$print file &

W zasadzie nie ma ograniczenia liczby poleceń wykonywanych w ten sposób (w tzw. tle).

W Unixie używa się niekiedy pojęcia metaznaku (ang. metacharacter), mówiąc o znakach mających specjalne znaczenie dla powłoki, tj.:„>", „<", „|", i „&"
a także o dwóch znakach służących do wyrażania zmiennych nazw, o czym była mowa w pierwszej części artykułu, czyli: „?" i „*".
Użycie tych znaków w ich znaczeniu literalnym, bez znaczenia przypisywanego im przez powłokę, jest możliwe przez poprzedzenie każdego z nich ukośnikiem (ang. backslash), zaliczanym także do metaznaków (nie można go użyć w tym artykule, ponieważ nie figuruje on w zestawie znaków drukarni składającej ten tekst).

PROGRAMOWANIE POWŁOKI

Wymienione dotąd możliwości powłoki są dosc proste i łatwo zrozumiałe dla czytelników mających już kontakt, na przykład, z systemem PC-DOS, który nawiasem mówiąc przejął je z Unixa. Omówione mechanizmy współpra, z systemem PC-DOS, który nawiasem mówiąc przejął je z Unixa. Omówione mechanizmy współpracy programów są jednak dość ograniczone. Bardziej złożone funkcje uzyskuje się metodami, które można zaliczyc do programowania w ścisłym sensie. Należy do nich:
– operowanie zmiennymi,
– tworzenie programów, czyli procedur powłoki (tzw. skryptów, ang. scripts),
– przekazywanie parametrów procedurom powłoki,
– użycie struktur sterujących,
– wykorzystanie poleceń wewnętrznych.

Zmienne

Powłoka ma kilka zmiennych o wartościach zdefiniowanych pierwotnie. Należą do nich m.in. zmienne *PS1* i *PS2* (znaki gotowości, ang. prompt strings), *HOME* (nazwa skorowidza macierzystego, tj. domyślny argument polecenia cd), *PATH* (zbiór skorowidzów przeszukiwanych przez system operacyjny w poszukiwaniu programów poleceń) i inne.

Wartości zmiennych zdefiniowanych pierwotnie można zmieniać. Przykładowo, zamiast typowych znaków gotowości systemu operacyjnego można wprowadzić nowe:

PS1 = słucham kochanie
PS2 = jeszcze

Wartość zmiennej *PS2* oznacza gotowość do przyjmowania dalszego ciągu tego samego polecenia , ale w nowym wierszu. Zmiennej *HOME* można nadać wartość:

HOME = /usr/zal

a zmiennej PATH – wartość
PATH = .:./bin:/bin:/usr/bin:$HOME/bin
Przeszukiwanie skorowidzów przez system operacyjny po wydaniu polecenia odbywa się w kolejności określonej przez zmienną *PATH* (poszczególne ścieżki są oddzielone dwukropkiem), a więc: skorowidz bieżący, skorowidz *bin* skorowidza bieżącego, systemowy skorowidz *bin*, skorowidz *bin* skorowidza /usr, *skorowidz bin* skorowidza *zal*.

Wartość zmiennej może być podstawiona za zmienną w wyrażeniu, jeśli poprzedzi się nazwę zmiennej znakiem „$", np. (jak powyżej):

$echo $HOME

zamiast

$echo /usr/zal

Polecenie *echo* powoduje wyprowadzenie na terminal całej informacji wprowadzonej w tej samej linii.

Wszystkie wartości zmiennych definiowanych pierwotnie można zapisać na pliku *.profile*, który powinien znajdować się w skorowidzu macierzystym użytkownika. Jest on odczytywany przez powłokę podczas rejestrowania się użytkownika w systemie.

Nazwy zmiennych stanowią ciągi znaków zaczynające się od litery i złożone z liter, cyfr i znaków podkreślenia. Wartościami zmiennych są napisy. Wartością zmiennej może byc też nazwa polecenia, ponieważ jest napisem, np.:

S = "sort"

Użycie takiego polecenia polega na wywołaniu wartości zmiennej, tj.:

$ $S

Skrypty

Zwykłe wydawanie pojedynczych poleceń z klawiatury terminala jest tyleż proste co męczące. Można je jednak zapisywać w języku powłoki tworząc pewien rodzaj programów. Programowanie powłoki polega przede wszystkim na grupowaniu poleceń, zapisywaniu ich na plik i wykonywaniu w trybie zwanym pośrednim lub wsadowym. Polecenia zgrupowane na pliku nazywa się procedurami powłoki, programami powłoki lub skryptami (ang. scripts) Sama zasada jest dość znana i szeroko stosowana w używanych w Polsce systemach operacyjnych: RSX-11M, dla komputerów zgodnych z PDP-11, i PC-DOS – dla IBM PC i kompatybilnych. W pierwszym wypadku są to tzw. pliki poleceń (ang. indirect command files, typu CMD), a w drugim pliki wsadowe (ang. batch files, typu BAT). W Unixie jednak możliwości programowania powłoki są znacznie większe – choć w tym artykule tylko zasygnalizowane.

Skryptem lub procedurą powłoki nazywa się program napisany w języku powłoki, tj. używający istniejących w systemie poleceń i reguł składniowych (omówionych poniżej). Przykładowo, jeżeli plik *file* zawiera ciąg poleceń:

cd ..
pwd
ls -l

to wykonanie polecenia

$sh file

polega na kolejnym wykonaniu wszystkich poleceń zapisanych na tym pliku (polecenie *sh* oznacza wywołanie powłoki, ang. shell).

Warto wiedzieć, że plik *file* można uczynić bezpośrednio wykonywalnym zmieniając odpowiednio jego prawa dostępu poleceniem *chmod* (znanym z trzeciego odcinka tego artykułu), tzn.:

$chmod 755 file

które nadaje plikowi atrybuty:

rwxr-xr-x

Po wykonaniu tego polecenia plik *file* można wywoływać bezpośrednio, tzn.

$file

tak jak każde polecenie.

Przekazywanie parametrów

Użyteczność skryptów, które pod różnymi nazwami są już dość szeroko stosowane w technice systemów operacyjnych, znacznie wzrasta dzięki możliwościom przekazywania parametrów. Parametry oznacza się symbolami *$1, ..., $9*, tzn. może ich być najwyżej 9

w jednym skrypcie. Ogólne wywołanie skryptu z parametrami ma postać:

$sh file param_1, ..., param_9

Wewnątrz skryptu poszczególne parametry muszą być zapisane w postaci *$1, ..., $9,* np. skrypt *file* o treści:

who | grep $1

ma następujące wywołanie:

$sh file param_1

Użyte tu polecenie *who* (po polsku – *kto*) powoduje wyprowadzenie danych o użytkownikach zarejestrowanych w systemie, a polecenie *grep* – wyszukanie w pliku wejściowym i wyprowadzenie wszystkich wystąpień określonych wyrażeń (ang. global regular expressions print).

Ograniczenie liczby parametrów do dziewięciu, choć dość sztuczne, nie jest zbyt istotne i można je ominąć. W programach można także używać nazwy skryptu udostępnianej przez zmienną $# 1 liczby parametrów udostępnianej przez zmienną $O.

Struktury sterujące

Struktury sterujące są bodaj najważniejszym elementem programowania powłoki, gdyż one w zasadzie tworzą język powłoki. Do stosowania tych konstrukcji istotna jest informacja, że każde polecenie ma tzw. stan końcowy (ang. exit status), sygnalizowany w wypadku powodzenia (wartość *TRUE*) lub niepowodzenia (wartość *FALSE*) przy realizacji polecenia. Korzystając z tego warunku można używać następujących konstrukcji:
1) tzw. koniunkcji warunkowej, symbolizowanej podwójnym znakiem &&:

polecenie_1 && polecenie_2

która polega na wykonywaniu *polecenia_2* tylko wtedy, gdy warunek (stan) koncowy *polecenia_1* ma wartość *TRUE,* tzn. gdy wykonanie *polecenia_1* zakończy się powodzeniem;
2) alternatywy warunkowej symbolizowanej dwoma belkami ||

polecenie_1 || polecenie_2

która polega na wykonywaniu *polecenia_2* tylko wtedy, gdy wykonanie *polecenia_1* zakończy się niepowodzeniem (stan końcowy *polecenia_1* przybierze wartość *FALSE*).

Stanem końcowym koniunkcji warunkowej i alternatywy warunkowej jest stan końcowy ostatniego wykonywanego polecenia. Tak więc, stan końcowy koniunkcji warunkowej ma wartość *TRUE*, jeżeli obydwa polecenia kończą się powodzeniem, natomiast stan końcowy alternatywy warunkowej ma wartość *TRUE*, jeżeli jedno z poleceń zakończy się powodzeniem.

Ważniejszą rolę odgrywają jednak polecenia warunkowe (rozgałęzienia) *if* i *case* oraz pętle *for* i *while*. Polecenie *if* ma następującą składnię:

```
if lista_poleceń
then lista_poleceń
else lista_poleceń
fi
```

przy czym część *else* jest opcjonalna. Jak łatwo zauważyć polecenie warunkowe *if* bez części *else* jest równoważne koniunkcji warunkowej.

Polecenie *case* ma następującą postac:

```
case wartość in
 wartość_1) lista_poleceń ;;
 ...
 wartość_n) lista_poleceń
esac
```

Dobrym przykładem ilustracji działania polecenia *case* jest skrypt polecenia *append* – dołączania pliku do pliku:

```
case $# in
 1) cat >> $1 ;;
 2) cat >> $2 <$1 ;;
*) echo 'append – błędne użycie polecenia'
esac
```

Użycie skryptu *append* ma postać:

$append file_1

lub

$append file_1 file_2

W pierwszym wypadku, ponieważ występuje tylko jeden argument, zmienna *$#* przybiera wartość *1* i wykonywana jest pierwsza ewentualność polecenia *case*, tj.

cat >> file_1

a więc zapisanie informacji na plik *file_1*. W drugim wypadku występują dwa argumenty, *file_1* i *file_2,* a więc wartość zmiennej $# wynosi *2* i wykonywana jest druga ewentualność polecenia *case*. Podanie każdej innej liczby argumentów powoduje wykonanie ewentualności oznaczonej gwiazdką „*" (jak pamiętamy, w Unixie gwiazdka może oznaczać dowolny napis).

Odpowiednie pętle oparte na badaniu warunku (pętla *while*) i odliczaniu wartości (pętla *for*) mają postać:

```
while lista_poleceń
do lista_poleceń
done
```

oraz

```
for zmienna in lista_wartości
do lista_poleceń
done
```

Przykładowy skrypt polecenia:

dw skorowidz

służącego do wyprowadzania całego wykazu (drzewa) skorowidzów i zawartych w nich plików ma postać:

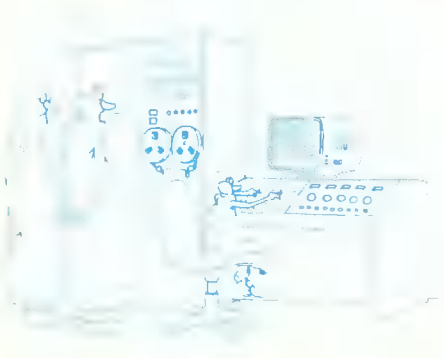
```
cd $1
ls -l
for i in *
do
 if test– f $i
 then :
 else dw $i
 fi
done
```

Znaczenie poleceń *cd* i *ls* powinno być łatwe do rozszyfrowania, podobnie jak znaczenie gwiazdki w linii polecenia *for* (gwiazdka oznacza w Unixie dowolny ciąg znaków, a więc w tym wypadku – pełen zakres wartości liczbowych) i sens tzw. rekurencyjnego wywoływania polecenia *dw* (wywołanie rekurencyjne oznacza wywołanie jakiegoś programu z niego samego; umożliwiają to specjalne techniki programowania). Jednak do pełnego wyjaśnienia powyższego skryptu konieczne jest omówienie użytych poleceń wbudowanych.

Polecenia wbudowane

Powłoka Unixa jest programem na tyle złożonym, że sama zawiera pewne polecenia wewnętrzne (wbudowane), tzn. niedostępne na zewnątrz niej, podobnie jak inne bardziej skomplikowane programy w każdym systemie operacyjnym, np. edytor, program uruchomieniowy itp.

Spośród kilkunastu poleceń wbudowanych powłoki, w poniższym przykładzie użyto dwóch:
– dwukropek „:", oznaczający polecenie puste (nic nie rób), udostępniające jedynie stan (zawsze *TRUE*),
– test, służące do badania określonych warunków, w tym wypadku warunku, czy określony plik o nazwie będącej argumentem wywołania (*$1*) istnieje i nie jest skorowidzem (polecenie test często występuje również jako nie wbudowane).



JANUSZ ZALEWSKI

ELEMENTY GRAFIKI W JĘZYKU TURBO-PASCAL

Bardzo przydatnym rozszerzeniem możliwości graficznych języka TURBO-Pascal jest jego biblioteka zawarta w zbiorze **Graph.bin**. Dostęp do niej łatwo uzyskać, dokonując w programie źródłowym wprowadzenia deklaracji ze zbioru **Graph.p**. Po wykonaniu tej prostej czynności można w programach posługiwac się procedurami do wykreślania łuków i okręgów, procedurami do wypełniania obszarow punktami o wybranym kolorze lub punktami tworzącymi zadany wzór, jak również odwoływać się do dość obszernego podzbioru procedur grafiki żółwiowej i procedur umożliwiających animację. Wraz z procedurami zintegrowanymi z językiem, a więc nie wymagającymi wprowadzenia, takimi jak procedury do wykreślania punktów i odcinków oraz procedurami do zarządzania trybami graficznymi i oknami, powstaje dostatecznie rozbudowany system współpracy z ekranem ułatwiający wykonywanie operacji graficznych.

Dla zilustrowania wybranych możliwości graficznych języka TURBO-Pascal na wydruku 1 przedstawiono tekst programu do projektowania wzorów. W programie tym można zauważyć dyrektywę wprowadzenia zbioru **Graph.p** oraz dyrektywę wprowadzenia zbioru **Logo.sys**. Drugi z tych zbiorów, przytoczony na wydruku 2, zawiera procedury opatrywania wykresów inicjałami **jb**.

Jak wynika z opisu firmowego języka TURBO-Pascal w wersji dla IBM PC, wykonanie procedury:

Pattern (P)

gdzie P jest nazwą tablicy typu **array [0..7] of byte**, powoduje zdefiniowanie wzoru o rozmiarach 8x8 bitów.

Po zdefiniowaniu wzoru można posłużyc się procedurą:

FillPattern (xMin, yMin, xMax, yMax, Color)

której wykonanie spowoduje wypełnienie prostokąta o narożnikach **(xMin, yMin)** i **(xMax, yMax)** zdefiniowanym wzorem w kolorze **Color**. Cała trudność posłużenia się wymienionymi procedurami sprowadza się zatem do zdefiniowania wzoru, który na dodatek musi po zaprojektowaniu zostac lustrzanie odwzorowany w osi pionowej i poziomej. Zadanie to ma ułatwić omawiany program **PatternDesigner**.

Bezpośrednio po wywołaniu tego programu, na ekranie pojawia się szachownica o rozmiarach 8x8, po której można się dowolnie poruszać, posługując się ośmioma klawiszami kierunkowymi, w tym czterema klawiszami oznaczonymi strzałkami – w górę, w dół, w lewo, i w prawo. Nacisnięcie klawisza **Ins** powoduje zabarwienie kwadracika szachownicy, a naciśnięcie klawisza **Del** – jego odbarwienie.

Jednocześnie z wypełnianiem szachownicy zabarwionymi kwadracikami, z jej prawej strony, pojawiają się szesnastkowe kody poszczególnych jej elementów, a pod tymi kodami – wzór reprezentowany na szachownicy – zmniejszony do jednego znaku.
Poza opisanymi klawiszami, można posługiwać się klawiszami sterującymi **F1-F4**, którym przypisano następujące znaczenie:

F1 – zakończenie wykonywania programu,

F2 – wypełnienie zdefiniowanym wzorem dużego kwadratu zajmującego lewą część ekranu (rys. 1),

F3 – przywrócenie początkowych warunków wykonywania programu,

F4 – przedstawienie wzoru we wnętrzu okręgu (rys. 2).

Niciśnięcie dowolnego z pozostałych klawiszy nie ma żadnych skutków.

Pozostawiając czytelnikom samodzielne przestudiowanie programu, który – moim zdaniem – stanowi dobrą ilustrację możliwości programowych oferowanych przez język TURBO-Pascal, pozostaje jedynie wyjaśnić rolę jaką pełni ciąg szesnastkowych liczb dwucyfrowych wymienionych w nawiasach u dołu ekranu. Jest to właśnie ten ciąg liczb szesnastkowych, które należy przypisać argumentowi procedury **Pattern**, aby po wywołaniu procedury **FillPattern** uzyskać wzór zaprojektowany na szachownicy.

Aby ułatwic analizę programu **PatternDesigner**, przytaczam krótkie opisy użytych w nim procedur graficznych. Tam gdzie jest mowa *o* kolorze, obowiązują następujące domniemania palety barw: **0**– czarna, **1**– zielona, **2** – czerwona, **3** – brązowa.

Procedura GraphColorMode

Wywołanie: **GraphColorMode**

Wykonanie procedury powoduje ustanowienie trybu graficznego – barwnego 320x200 pikseli.

Procedura GraphWindow

Wywołanie: **GraphWindow (xMin, yMin, xMax, yMax)**

Wykonanie procedury powoduje zdefiniowanie prostokątnego okna graficznego. **(xMIn, y,Min)** i **(xMax, yMax)** są współrzędnymi przeciwległych wierzchołków prostokąta.

Procedura ClearScreen

Wywołanie: **ClearScreen**

Wykonanie procedury powoduje wyzerowanie okna graficznego.

Procedura Plot

Wywołanie: **Plot (xCor, yCor, Color)**

Wykonanie procedury powoduje wyświetlenie w punkcie okna o współrzędnych **(xCor, yCor)**, jednego piksela w kolorze **Color**.

Procedura Draw

Wywołanie: **Draw (xMin, yMin, xMax, yMax, Color)**

Wykonanie procedury powoduje wyświetlenie w oknie, między punktami o współrzędnych **(xMin, yMin)** i **(xMax i yMax)**, odcinka linii prostej, w kolorze **Color**.

Procedura Circle

Wywołanie: **Circle (xCor, yCor, Radius, Color)**

Wykonanie procedury powoduje wykreslenie okręgu o środku w punkcie o współrzędnych **(xCor, yCor)**, promieniu **Radius** i kolorze **Color**.

Procedura Pattern

Wywołanie: **Pattern (Patt)**

Wykonanie procedury Patt powoduje zdefiniowanie wzoru do wypełniania obszarów. **Patt** reprezentuje tablicę typu **array [0..7] of byte,** określającą wzór 8x8 pikseli

Procedura FillPattern

Wywołanie: **Fillpattern (xMin, yMin, xMax, yMax, Color)**

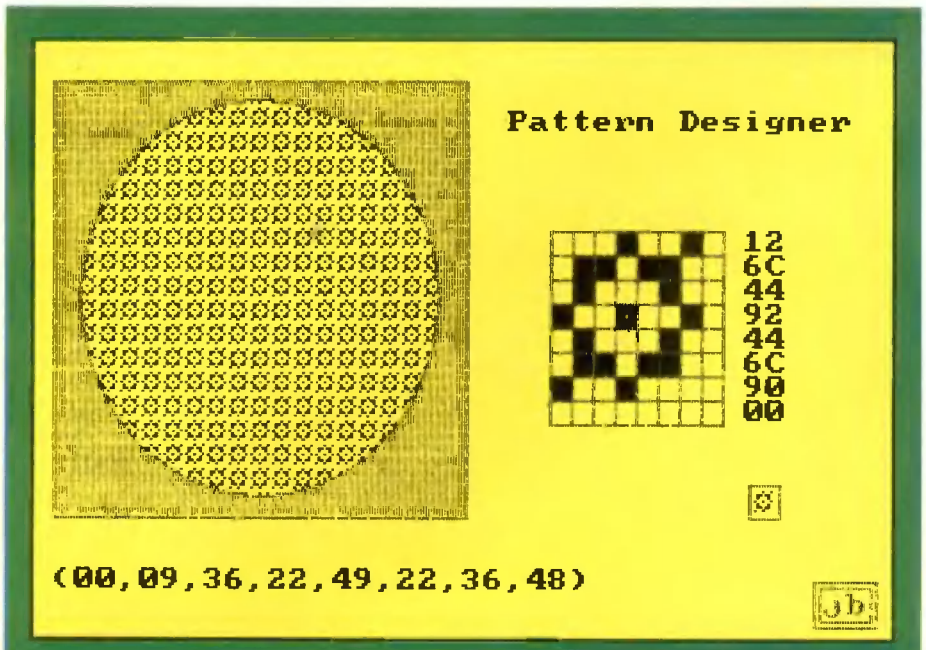
Wykonanie procedury powoduje wypełnie-

```
(1)   program PatternDesigner;

(2)   {$i Graph.p  }
(3)   {$i Logo.sys }

(4)   type
(5)     PatternType = array[0..7] of byte;

(6)   const
(7)     xMin =   8; yMin =  14 ;
(8)     xMax = 161; yMax = 159;
(9)     xPad =  25; yPad =   9;
(10)    HexDigits : string[16] = '0123456789ABCDEF';
(11)    Hor = 319;
(12)    Ver = 199;
(13)    EmptyPattern : PatternType =
(14)                     (0,0,0,0,0,0,0,0);
(15)  var
(16)    PatternArray,Vector : PatternType;
(17)    Row : byte;
(18)    x,y,xPix,yPix,i,j : integer;
(19)    Ch : char;
(20)    Finish : boolean;

(21)  procedure DrawBorder(xMin,yMin,
(22)                       xMax,yMax : integer;
(23)                       Color : byte );
(24)  begin
(25)    Draw(xMin,yMin,xMax,yMin,Color);
(26)    Draw(xMax,yMin,xMax,yMax,Color);
(27)    Draw(xMax,yMax,xMin,yMax,Color);
(28)    Draw(xMin,yMax,xMin,yMin,Color)
(29)  end;

(30)  procedure DrawGrid;
(31)  begin
(32)    xPix := (xPad - 1) * 8;
(33)    yPix := (yPad - 1) * 8;
(34)    for i := 0 to 8 do begin
(35)      Draw(xPix,yPix + 8 * i,
(36)           xPix + 64,yPix + 8 * i,2);
(37)      Draw(xPix + 8 * i,yPix,
(38)           xPix + 8 * i,yPix + 64,2);
(39)      if i < 8 then begin
(40)        gotoXY(xPad + 9,yPad + i);
(41)        Write('00')
(42)      end
(43)    end
(44)  end;

(45)  procedure InvertPixel;
(46)  var
(47)    Pixel : integer;
(48)  begin
(49)    xPix := (xPad - 1) * 8 + x * 8 + 4;
(50)    yPix := (yPad - 1) * 8 + y * 8 + 4;
(51)    Pixel := GetDotColor(xPix,yPix);
(52)    Plot(xPix,yPix,3 - Pixel)
(53)  end;

(54)  procedure ChangePad(Insert : boolean);
(55)  var
(56)    Mask : byte;
(57)  begin
(58)    Mask := 128 shr x;
(59)    Row := PatternArray[y];
(60)    Row := Row and not Mask;
(61)    if Insert then
(62)      Row := Row or Mask;
(63)    PatternArray[y] := Row;
(64)    xPix := (xPad - 1) * 8 + x * 8 + 2;
(65)    yPix := (yPad - 1) * 8 + y * 8 + 2;
(66)    FillShape(xPix,yPix,0,2);
(67)    if Insert then
(68)      FillShape(xPix,yPix,3,2);
(69)    gotoXY(xPad + 9,yPad + y);
(70)    Write(HexDigits[Row shr 4  + 1],
(71)          HexDigits[Row and $F + 1]);
(72)  end;

(73)  procedure DisplayVector;
(74)  begin
(75)    gotoXY(2,23);
(76)    Write('(');
(77)    for j := 0 to 7 do begin
(78)      Row := Vector[j];
(79)      Write(HexDigits[Row shr 4  + 1],
(80)            HexDigits[Row and $F + 1]);
(81)      if j < 7 then Write(',')
(82)    end;
(83)    Write(')')
(84)  end;

(85)  procedure DrawPattern;
(86)  const
(87)    xPix = 266;
(88)    yPix = 148;
(89)  var
(90)    x,y : integer;
(91)  begin
(92)    DrawBorder(xPix,yPix,xPix + 11,yPix + 11,2);
(93)    for y := 0 to 7 do
(94)      for x := 0 to 7 do
(95)        Plot(xPix + 2 + x,yPix + 2 + y,
(96)             3 * (ord(PatternArray[y] and
(97)                  (128 shr x) <> 0)));
(98)  end;
(99)  procedure FillSquare;
(100) begin
(101)   GraphWindow(xMin + 1,yMin + 1,
(102)                xMax - 1,yMax - 1);
(103)   ClearScreen;
(104)   GraphWindow(0,0,Hor,Ver);
(105)   for i := 0 to 7 do begin
(106)     Row := PatternArray[7 - i];
(107)     for j := 0 to 7 do begin
(108)       Vector[i] := Vector[i] shl 1 + Row and 1;
(109)       Row := Row shr 1
(110)     end
(111)   end;
(112)   DisplayVector;
(113)   Pattern(Vector);
(114)   FillPattern(xMin + 1,yMin + 1,
(115)                xMax - 1,yMax - 1,3)
(116) end;

(117) procedure Trim;
(118) begin
(119)   Circle((xMin + xMax) shr 1,
(120)          (yMin + yMax) shr 1,
(121)          (xMax - xMin) shr 1 - 10,2);
(122)   FillShape(xMin + 1,yMin + 1,1,2)
(123) end;

(124) procedure Initialize;
(125) begin
(126)   ClearScreen;
(127)   jbLogo;
(128)   DrawBorder(0,0,Hor,Ver,3);
(129)   DrawBorder(xMin,yMin,xMax,yMax,2);
(130)   DrawGrid;
(131)   gotoXY(23,4);
(132)   Write('Pattern Designer');
(133)   x := 0;
(134)   y := 0;
(135)   PatternArray := EmptyPattern;
(136)   InvertPixel
(137) end;

(138) begin

(139)   GraphColorMode;

(140)   Initialize;

(141)   Finish := false;
(142)   repeat
(143)     DrawPattern;
(144)     Read(Kbd,Ch);
(145)     if (Ch = ^[) and KeyPressed then begin
(146)       Read(Kbd,Ch);
(147)       InvertPixel;
(148)       case ord(Ch) of
(149)         75 : { W }   x := x - 1;
(150)         77 : { E }   x := x + 1;
(151)         72 : { N }   y := y - 1;
(152)         80 : { S }   y := y + 1;
(153)         71 : { NW } begin
(154)                      y := y - 1;
(155)                      x := x - 1
(156)                    end;
(157)         73 : { NE } begin
(158)                      y := y - 1;
(159)                      x := x + 1
(160)                    end;
(161)         79 : { SW } begin
(162)                      y := y + 1;
(163)                      x := x -1
(164)                    end;
(165)         81 : { SE } begin
(166)                      y := y + 1;
(167)                      x := x + 1
(168)                    end;
(169)         82 : { Ins } ChangePad(true);
(170)         83 : { Del } ChangePad(false);
(171)         62 : { F4 }  Trim;
(172)         61 : { F3 }  Initialize;
(173)         60 : { F2 }  FillSquare;
(174)         59 : { F1 }  Finish := true
(175)       end;
(176)       x := x and 7;
(177)       y := y and 7;
(178)       if ord(ch) <> 61 then InvertPixel
(179)     end
(180)   until Finish;

(181)   TextMode

(182) end.
```

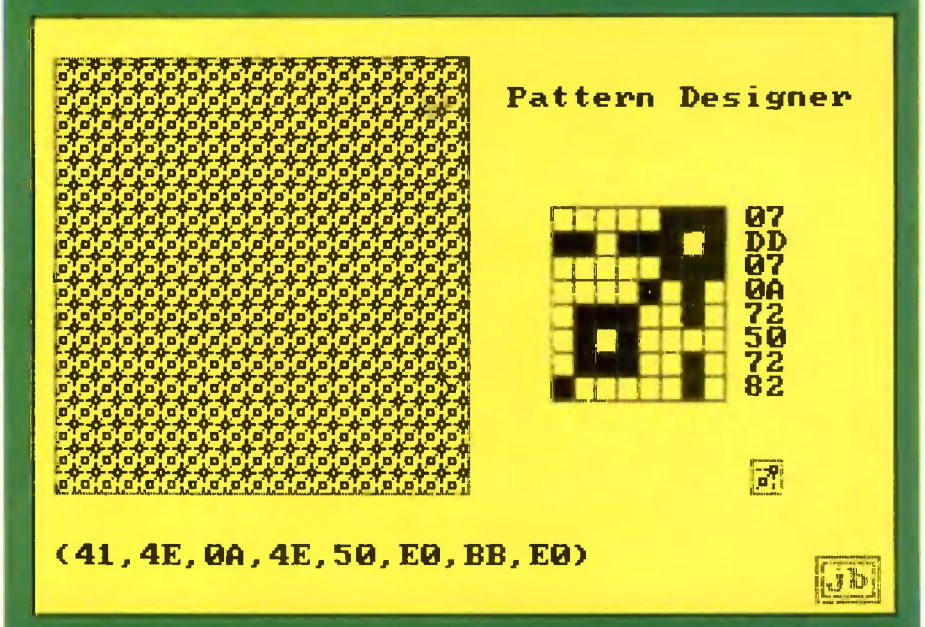
Wydruk 1. Program do projektowania wzorów

```
{ Logo.sys }

procedure Logo(xLogo,yLogo : integer);

const
  Logo : array[0..15,0..2] of byte =

         (($FF,$FF,$FF),($80,$00,$01),
          ($BF,$FF,$FD),($A0,$00,$05),
          ($A0,$BE,$05),($A0,$D6,$05),
          ($A1,$C7,$C5),($A0,$C6,$65),
          ($A0,$C6,$65),($AC,$C6,$65),
          ($AC,$CF,$C5),($A7,$B0,$05),
          ($A0,$00,$05),($BF,$FF,$FD),
          ($B0,$00,$01),($FF,$FF,$FF));
var
  OneByte,x,y,i : byte;

begin
  for y := 0 to 15 do
    for x := 0 to 2 do begin
      OneByte := Logo[y,x];
      for i := 0 to 7 do
        Plot(xLogo + x * 8 + i,yLogo + y,
             ord((OneByte and (128 shr i)) <> 0) shl 1)
    end
end;

procedure jbLogo;
begin
  Logo(290,180)
end;
```

Wydruk 2. Procedury do generowania inicjałów



Rys. 2. Wypełnienie wzorem okręgu



Rys. 1. Wypełnienie wzorem kwadratu

nie prostokąta punktami o kolorze **Color**. **(xMin, yMin) i (xMax, yMax)** są współrzędnymi przeciwległych wierzchołków prostokąta.

Procedura FillShape

Wywołanie: **FillShape (xCor, yCor, Fill, Border)**

Wykonanie procedury powoduje zlokalizowanie punktu okna o współrzędnych **(xCor, yCor)**, a następnie wypełnienie obszaru otaczającego ten punkt – punktami w kolorze **Fill**. Zakłada się, że obrzeże obszaru otaczającego wspomniany punkt ma kolor **Border**.

JAN BIELECKI

JAK PISAĆ DO MIKROKLANU...

MIKROKLAN jest dla Ciebie. Oznacza to, że i od Ciebie zależy, co będzie w nim drukowane. Będziemy wdzięczni, jeżeli napiszesz dlaczego Ci się to czy tamto nie podoba, a szczególnie wdzięczni, gdy zaproponujesz co zrobić, żeby było lepiej. Pamiętaj jednak, że papier nie jest z gumy i mieści się na nim skończona liczba informacji. Weź też pod uwagę prawdopodobną opinię pozostałych 99 999 Czytelników (niektórzy spośród nich są bez mała ekspertami, inni zaś stawiają pierwsze kroki...). Nadsyłane nam propozycje tematów, które zdaniem Czytelników powinny znaleźć się na łamach MIKROKLANU, kwalifikować będziemy według liczby zainteresowanych. Może właśnie Twój list przeważy szalę? Będziemy starali się reagować jak najszybciej, jednak droga od pomysłu do ukazania się na rynku zajmuje kilka miesięcy.

Jeśli uważasz, że wiesz coś ciekawego i chciałbyś podzielić się tym z pozostałymi Czytelnikami – napisz artykuł. Jak wskazuje nazwa naszego pisma, pragniemy tworzyć wspólnie z Czytelnikami klan ludzi, którym zależy, aby mikrokomputery były wykorzystywane z pełną świadomością ich możliwości i ograniczeń. Możesz więc napisać o tym, co sam odkryłeś lub udoskonaliłeś, posługując się Spectrum, Commodorem, Amstradem czy IBM PC (jeśli zainspirował Cię pomysł już gdzieś publikowany, powołaj się na niego, załączając do tekstu wykaz literatury). Możesz napisać o możliwościach rozbudowy lub bardziej efektywnym sposobie programowania. Możemy też opublikować Twój program, jeśli będzie on interesujący dla innych i... niezbyt długi. To samo dotyczy konstrukcji sprzętowych. Ciekawe rozwiązania układowe, niestandardowe aplikacje, czyli forum do wymiany pomysłów i doświadczeń dla wszystkich, hobbistów i profesjonalistów.

Wymagania dotyczące przysyłanych materiałów:

tekst: w maszynopisie, z dwoma kopiami, (30 wierszy na stronie, 60 uderzeń w wierszu, czyli duży odstęp między liniami)

objętość: maksymalnie 7-9 stron maszynopisu (jasno i zwięźle)

rysunki: mogą być w ołówku (przejrzyste i czytelne)

wydruki: maksymalnie kontrastowe (załóż nową taśmę do drukarki, jeśli możesz przyślij program na nośniku magnetycznym – zwrócimy!)

programy: nie tylko przetestowane, ale też eleganckie

konstrukcje: wyłącznie uruchomione i koniecznie sprawdzone

ponadto: adres domowy, telefony – domowy i służbowy, ew. numer konta PKO.

UKŁAD INTERFEJSU DO ATARI 600XL I 800XL

Duża popularność komputerów domowych ATARI 600XL i 800XL oraz fakt, że stały się one dostępne w sklepach Pewexu, zachęciło autorów do opracowania interfejsu przeznaczonego do współpracy komputera z magnetofonem kasetowym. Na giełdach pojawiać zaczęły się ostatnio podobne interfejsy w cenie ok. 10 tys. zł. Koszt proponowanego urządzenia wynosi ok. 2 tys. zł i jest uzależniony od ceny układu NE565. Zalety proponowanego rozwiązania to zastosowanie tylko dwóch układów scalonych, bardzo prosta aplikacja oraz możliwość pracy na oryginalnie nagranych taśmach. Wadą – zastosowanie układu NE565 firmy Signetics oraz wynikająca stąd konieczność symetrycznego zasilania ±5 V (z komputera dostępne jest tylko +5V).

FSK – MODULACJA Z KLUCZOWANIEM CZĘSTOTLIWOŚCI

Do przesyłania danych z małymi i średnimi częstotliwościami stosuje się modulację FSK. Polega ona na wytwarzaniu dwóch częstotliwości znamionowych f_0 i f_1, z których każda odpowiada jednej z dwu wartości sygnału danych.
Zaletami takiego systemu są:
- prosta budowa układów nadawczo-odbiorczych,
- możliwość pracy niesynchronicznej,
- dobra odporność na zakłócenia szumem białym oraz zakłócenia impulsowe,
- mała wrażliwość na zniekształcenia tłumieniowe i opóźnieniowe.

Przy szybkościach transmisji większych od 1800 bodów stosuje się modulację fazy, która jest odporniejsza na zakłócenia. W komputerach ATARI 600XL i 800XL zastosowano modulację FSK z $f_0 = 4.2$ *kHz*, $f_1 = 5.6$ *kHz*.

ODBIÓR SYGNAŁÓW FSK

Spośród wielu znanych metod odbioru sygnałów FSK należy wymienić:
- odbiór z dyskryminatorem częstotliwości,
- odbiór z filtrami dopasowanymi (magnetofon firmowy ATARI 1010),
- odbiór różnicowy, polegający na porównywaniu faz dwóch sąsiednich elementów w sygnale odebranym,
- odbiór z pętlą fazową, zastosowany w niniejszym rozwiązaniu (rys. 1).

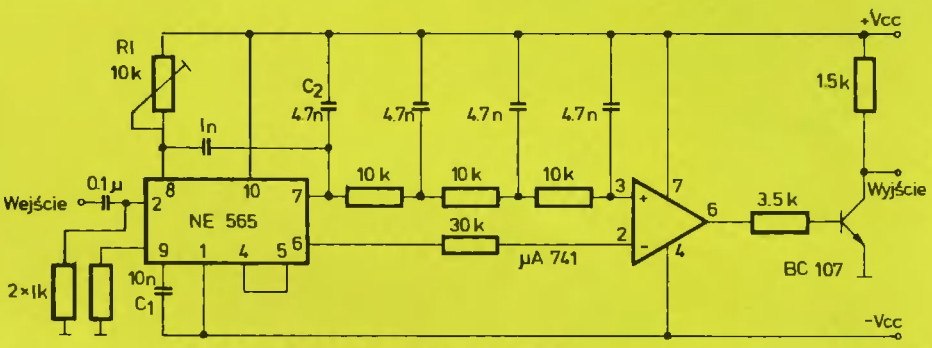
STROJENIE UKŁADU

Idea działania układu z wykorzystaniem PLL wydaje się tak prosta, że autorzy zrezygnowali z opisu. Zainteresowanych budową NE565 odsyłamy do literatury, podając jedynie wzory obliczeniowe:
- częstotliwość swobodna
  $f_0 = 1.2/4R_1C_1$ (Hz)
- zakres trzymania
  $f_L = \pm 8f_0/V_{CC}$ (Hz)
- zakres chwytania
  $f_C = \pm 1/2\pi \sqrt{2\pi f_L/\tau}$
  gdzie $\tau = 3.6 \star 10^3 \star C_2$

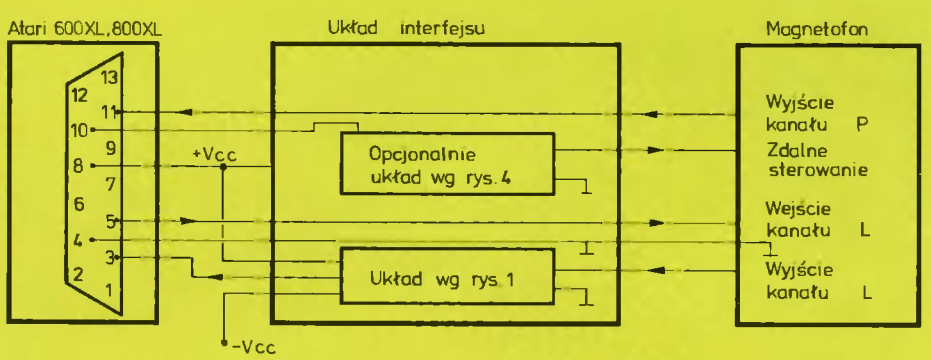
Pojemność C_2 filtru pętli wybrano mniejszą niż wynika z obliczeń dla eliminacji chwilowych przesterowań. Krawędzie pasma filtra łańcuchowego dobrano w przybliżeniu w połowie między maksymalnym zakresem kluczowania a podwojoną częstotliwością wejściową. Wartość R dobieramy tak, aby dla f_{in} = f_0 na wyjściu pętli uzyskać niewielkie dodatnie napięcie.

DOŁĄCZENIE INTERFEJSU

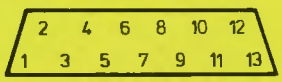
Sposób dołączenia interfejsu (rys. 2) jest bardzo prosty i wynika z opisu wyprowadzeń gniazda komputera (rys. 3). Komputery ATARI 600XL i 800XL przy współpracy z magnetofonem wykorzystują linie: *3, 4, 5, 8, 11*.



Rys. 1. Schemat demodulatora FSK dla ATARI 600XL i 800XL

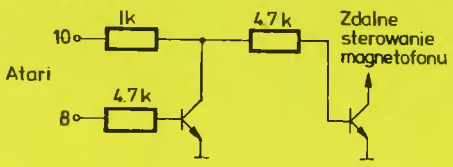


Rys. 2. Sposób podłączenia interfejsu (współpraca z magnetofonem monofonicznym poprzez kanał L)



Rys. 3. Opis wyprowadzeń gniazda komputera ATARI 600XL i 800XL (widok od strony wyprowadzeń)

1. Clock input	8. Motor controler
2. Clock output	9. Proceed
3. Data input	10. +5 V
4. Ground	11. Audio input
5. Data output	12. +12 V
6. Ground	13. Interrupt
7. Command	



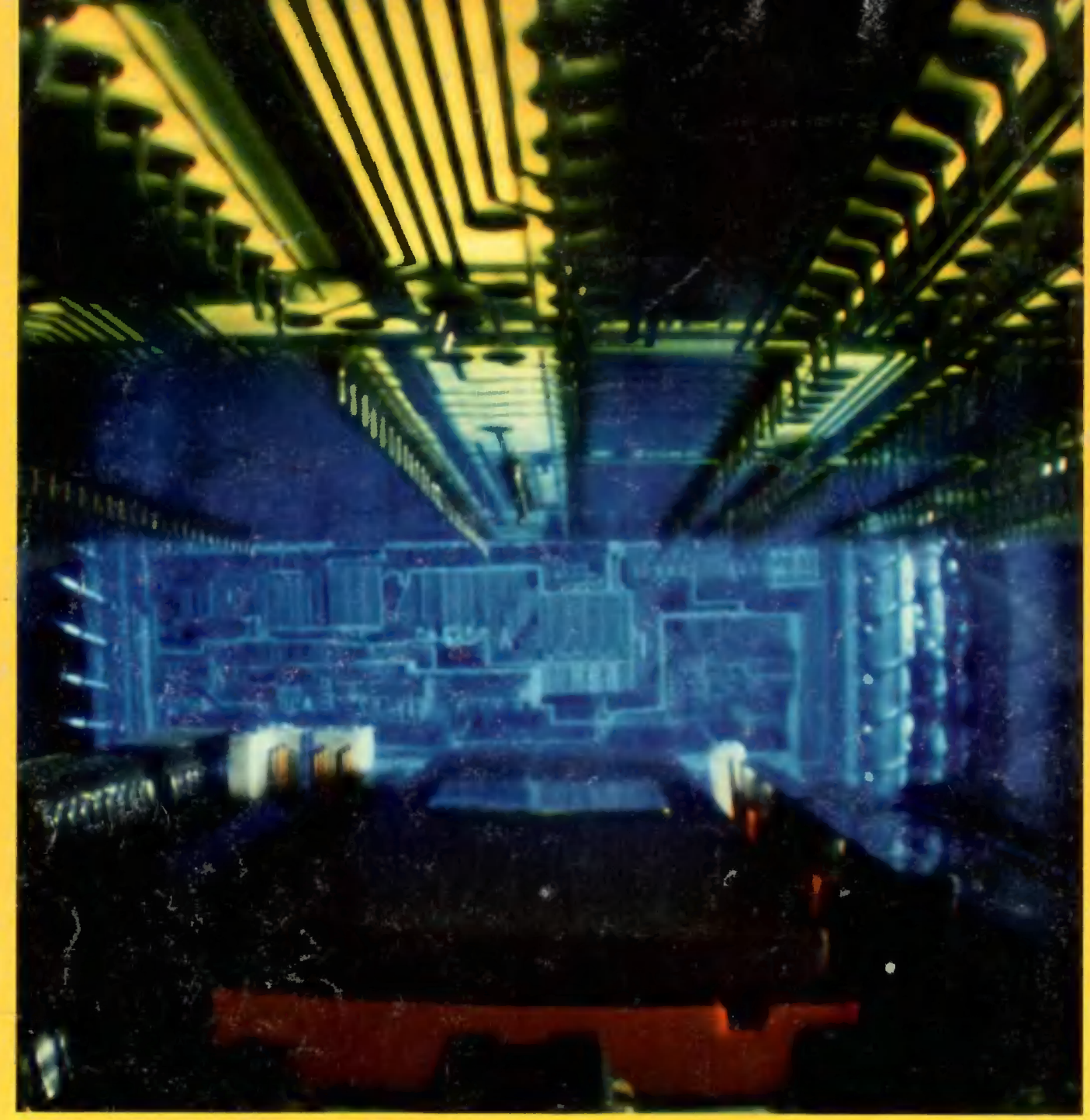
Rys. 4. Układ zdalnego sterowania magnetofonem

* * *

Końcówka 8 (Motor controler) służy do zasilania interfejsu oraz zdalnego sterowania silnikiem magnetofonu. Niektóre magnetofony, np. RMS 451, mają możliwość zdalnego sterowania zarówno przy odczycie, jak i zapisie, inne tylko przy zapisie, np. MK 235. Niewielka przeróbka MK 235 umożliwia sterowanie zarówno przy zapisie jak i odczycie.

Jeżeli magnetofon ma możliwość zdalnego sterowania, wówczas można dobudować układ sterujący magnetofonem (rys. 4). Wyprowadzenie 11 (rys. 2) jest wykorzystane do odtwarzania komentarza słownego nagranego na drugiej ścieżce magnetofonu stereofonicznego.

MIECZYSŁAW SZCZĘCH
KRZYSZTOF SKONIECZNY



W następnym zeszycie

o mikroprocesorach serii MC68000